

Machine Learning Introduction

DESY Summer Program 2025

July 2025

Notation

This section defines the notation used through the lecture course.

Numbers and Linear Algebra

a ,	A scalar value, such as a real number or integer
\mathbf{a} ,	A vector spanning a n -dimensional vector space
\mathbf{A} ,	A matrix, occupying the same n -dimensional vector space
\mathbf{A} ,	A tensor
\mathbf{I} ,	Identity matrix with implied dimension
$e^{(i)}$,	Standard basis vector with unit length at position i , $[0,0,1,0,0]$ for $i = 3$
a ,	A scalar random variable
\mathbf{a} ,	A vector-valued random variable
\mathbf{A} ,	A matrix-valued random variable

Sets and Graphs

\mathbb{A} ,	A set
\mathbb{R} ,	Set of real values
$\{0, 1\}$,	The set containing 0 and 1
$\{0, 1, \dots, n\}$,	The set of all integers between 0 and n
$[a, b]$,	Real interval including a and b
(a, b) ,	Real interval excluding a and b start/end points

Linear Algebra Operations

\mathbf{A}^T ,	Trasnpose of a matrix
$A \odot B$,	Element wise product
$\det(A)$,	Determinant of matrix \mathbf{A}

Calculus

$\frac{dy}{dx},$	Derivative of y with respect to x
$\frac{\partial y}{\partial x},$	Partial derivative of y with respect to x
$\nabla_x y,$	Gradient of y with respect to x
$\nabla_{\mathbf{X}} y,$	Matrix derivatives of y with respect to X
$\frac{f(\mathbf{x})}{d\mathbf{x}},$	Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Probability/Information Theory

$a \perp b,$	The two random variables are independent
$a \perp b c,$	The two random variables a and b conditioned on c, are independent
$P(a),$	Discrete probability distribution
$p(a),$	Continuous probability density function
$a \sim P,$	The value 'a' is a realisation (sampled) of distribution P
$\mathbb{E}_{x \sim P}[f(x)],$	Expectation of f(x) with respect to P(x)
$Var(f(x)),$	Variance of f(x) under the distribution P(x)
$Cov(f(x), g(x)),$	Covariance of f(x) and g(x) under P(x)
$\langle \rangle,$	

1 Linear Algebra

Linear algebra is the branch of mathematics concerned with linear equations and maps:

$$(x_1, x_2, \dots, x_n) \mapsto a_1 x_1 + a_2 x_2 + \dots + a_n x_n = b, \quad (1)$$

where x_i is known as the '*variable*', via their representation in vectors spaces. These concepts are expressed via vectors and matrix forms for easy manipulation and calculation. The following chapter is merely a review of linear algebra, which you are free to skip. If you are to skip this chapter, but would like a *cheat sheet* to quickly look up the mathematical language and/or concepts, then I would recommend *The Matrix Cookbook* by Peterson and Pederson (2006) [1].

1.1 Scalars, Vectors, Matrices, and Tensors

In linear algebra there are several types of mathematical constructs/objects:

- **Scalars:** A singular number represented by italics, defined by the type of number: $[\mathbb{N}, \mathbb{R}, \mathbb{Z}, \mathbb{C}]$. Which represent the natural set of positive integer, real, all integer, and complex numbers

- **Vectors:** An array of numbers, indexed by their ordering (x_i):

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}, \quad (2)$$

which space a n -dimensional. Each element depicts a location in the corresponding vector space, denoted by the scalar values type; e.g. the above vector $\mathbf{x} \in \mathbf{R}^n$ occupies a vector space in the real n -dimensional space.

Note 1

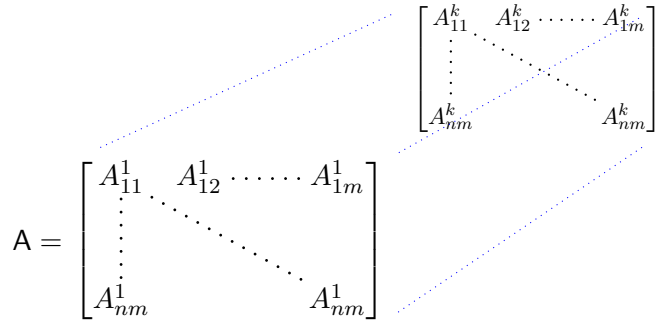
The symbol \in represents the element of a set.

- **Matrices:** A matrix \mathbf{A} is a 2-D array of numbers, meaning each element is indexed by two values ($A_{i,j}$). They therefore occupy the space defined by the joint product of the vector spaces. For example is $\mathbf{A} \in \mathbb{R}^{n \times m}$, where n/m are the dimensionality of the two vector spaces comprising the matrix:

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdot & \cdot & \cdot & A_{1,m} \\ A_{2,1} & A_{2,2} & \cdot & \cdot & \cdot & A_{2,m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ A_{n,1} & A_{n,2} & \cdot & \cdot & \cdot & A_{n,m} \end{bmatrix}. \quad (3)$$

Similar to how indexing works in common machine learning toolkits, we can index rows (1st row) and columns (1st column) via $\mathbf{A}_{1,:}$, and $\mathbf{A}_{:,1}$ respectively.

- **Tensors:** Similar to a matrix, but with more than two axes, thereby spanning a vector space of $n \times m \times \dots \times k$. The tensor is denoted by the typeface \mathbf{A} , and the element as $A_{i,j,k}$:



$$\mathbf{A} = \begin{bmatrix} A_{11}^1 & A_{12}^1 & \dots & A_{1m}^1 \\ \vdots & \vdots & \ddots & \vdots \\ A_{nm}^1 & \dots & \dots & A_{nm}^1 \end{bmatrix}$$

where in this example the tensor occupies $\mathbf{A} \in \mathbb{R}^{n \times m \times k}$ vector space.

Transpose One relevant mathematical operation on vectors and matrices is the transpose. The transpose operation of a matrix is the equivalent of a mirror image of the matrix across a diagonal line. Denoted by superscript T , the transpose is given by:

$$\mathbf{A}^T_{i,j} = \mathbf{A}_{j,i}, \quad (4)$$

for matrices, and for vectors is given by:

$$\mathbf{x}^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^T = [x_1, x_2, \dots, x_n], \quad (5)$$

which is essentially the process of swapping a column vector into a row vector.

Scalar, Vector, and Matrix Addition Provided that the vectors/matrices have the same shape, then addition is defined by the element wise addition:

$$\mathbf{C} = \mathbf{A} + \mathbf{B}, \quad (6)$$

where $C_{i,j} = A_{i,j} + B_{i,j}$. Similarly, a vector is also defined via the element wise addition, $\mathbf{z} = \mathbf{x} + \mathbf{y}$, where $z_i = x_i + y_i$. Since vectors/matrices occupy a vector space, scalar addition is also defined as the element wise addition: $\mathbf{C} = \mathbf{A} + b$ where $C_{i,j} = B_{i,j} + b$.

1.2 Vector and Matrix Multiplication

The multiplication of two matrices $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{l \times p}$ is possible if $m = l$, such that the new matrix $\mathbf{C} = \mathbf{AB}$, is given by the production operation:

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}, \quad (7)$$

which now occupies a vector space of dimensionality $\mathbb{R}^{n \times p}$.

This is in contrast to the *element wise product*, or *Hadamard product*, which is denoted as $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$, which is defined as:

$$C_{i,j} = A_{i,j} B_{i,j}, \quad (8)$$

where $i \in \mathbb{R}^n$ and $j \in \mathbb{R}^p$. This means that the Hadamard product is only defined for matrices of the same shape.

Note 2

In the context of your machine learning tool kits, such as PyTorch/Tensorflow/JAX/..., the idea of matrix-vector addition is defined in contrast to mathematics. Specifically, the addition of a vector $\mathbf{a} \in \mathbb{R}^n$ and matrix $\mathbf{A} \in \mathbb{R}^{m \times k}$, to form a new matrix \mathbf{B} :

$$\mathbf{B} = \mathbf{A} + \mathbf{a}$$

$$= \begin{bmatrix} A_{1,1} & A_{1,2} & \cdot & \cdot & \cdot & A_{1,k} \\ A_{2,1} & A_{2,2} & \cdot & \cdot & \cdot & A_{2,k} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ A_{n,1} & A_{n,2} & \cdot & \cdot & \cdot & A_{n,k} \end{bmatrix} + \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

$$= \begin{bmatrix} A_{1,1} & A_{1,2} & \cdot & \cdot & \cdot & A_{1,k} & a_{1,k+1} \\ A_{2,1} & A_{2,2} & \cdot & \cdot & \cdot & A_{2,k} & a_{2,k+1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ A_{n,1} & A_{n,2} & \cdot & \cdot & \cdot & A_{n,k} & a_{n,k+1} \end{bmatrix}.$$

This results in a new matrix $\mathbf{B} \in \mathbb{R}^{m \times (k+1)}$, and is only possible if $n = m$. This is often referred to as *broadcasting*, and is a purely deep learning concept not founded in the paradigm (theoretical framework) of mathematics.

Vector multiplication, or *inner dot product*, is defined between two vectors of the same dimension (\mathbb{R}^n). Such that, the new vector:

$$\mathbf{z} = \mathbf{x} \cdot \mathbf{y}, \quad (9)$$

is given by the element wise product $z_i = x_i y_i$. In contrast, the *outer dot product*, between two vectors of dimensionality \mathbb{R}^n and \mathbb{R}^m , creates a new matrix:

$$\mathbf{A} = \mathbf{x} \otimes \mathbf{y} \quad (10)$$

$$= \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdot & \cdot & \cdot & x_1 y_m \\ x_2 y_1 & x_2 y_2 & \cdot & \cdot & \cdot & x_2 y_m \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_n y_1 & x_n y_2 & \cdot & \cdot & \cdot & x_n y_m \end{bmatrix}. \quad (11)$$

where $\mathbf{A} \in \mathbb{R}^{n \times m}$. The resulting matrix is given by the operation $A_{i,j} = x_i y_j$.

Note 3

The Hadamard product is particularly useful for machine learning, in domains such as image processing, and time series analysis using recurrent neural networks (e.g. Long-Short-Term Memory (LST) and Gated Recurrent Units (GRU)). It is also a computational efficient operation - bit of foreshadowing here.

1.2.1 Matrix Operation Properties

Matrix multiplication is:

- Distributive:

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}, \quad (12)$$

- Associative:

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}. \quad (13)$$

However, matrix multiplication is not commutative:

$$\mathbf{AB} \neq \mathbf{BA}, \quad (14)$$

resulting in the transpose property of matrix products of:

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T. \quad (15)$$

1.3 Identity and Inverse

: Vector/Matrix notation is a powerful tool to express linear systems, as given the title of this chapter. Specifically, for a system defined by a set of linear equations, such as that of equation 1, the matrix formulation is simply:

$$\mathbf{Ax} = \mathbf{b}. \quad (16)$$

Solving this equation amounts to calculating the elements of \mathbf{A} , which can be achieved by *matrix inversion*. This requires the *identity* matrix \mathbf{I} , a matrix that leaves a vector unchanged, and preserves the n -dimensionality of said vector. Formally, $\mathbf{I} \in \mathbb{R}^{n \times n}$, and:

$$\forall x \in \mathbb{R}^n, \mathbf{Ix} = \mathbf{x}, \quad (17)$$

allowing us to defined the inverse of a matrix \mathbf{A} as:

$$\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}. \quad (18)$$

Consequently, solving equation 16 can be achieved by the following steps:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (19)$$

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (20)$$

$$\mathbf{I}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (21)$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \quad (22)$$

Of course this process depends on the existence of the inverse of \mathbf{A} , where the necessary conditions for said existence are discussed in the following section.

Note 4

Multiple algorithms exist for finding the inverse matrix \mathbf{A}^{-1} , which can be used to solve the equation multiple times for different \mathbf{b} vectors (with different values). Unfortunately, on a digital computer we can only represent \mathbf{A}^{-1} with finite precision, and so in software applications often algorithms that make use of \mathbf{b} obtain better precision.

1.4 Linear Dependence and Span

For the inverse matrix \mathbf{A}^{-1} to exist, the linear equation 1, must have a single solution for every value of \mathbf{b} . To analyse how many solutions the equation has, recall that the columns of \mathbf{A} specify the directions we can travel in the vector space from the origin (vector of all zero values), therefore we can see how many ways a value of \mathbf{b} can be reached by some combination of vectors. With this perspective each element of \mathbf{x} defines the amount one moves in the direction of the vector space basis i , such that:

$$\mathbf{A}\mathbf{x} = \sum_i x_i \mathbf{A}_{:,i}. \quad (23)$$

This operation is more formally known as a *linear combination*, which for a set of vector $\{\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(n)}\}$ is given by multiplying each vector by a scalar coefficient:

$$\sum_i c_i \mathbf{v}^{(i)}. \quad (24)$$

The *span* of a set of vectors is the set of all points reachable by the linear combination of basis vectors. Determining therefore whether $\mathbf{A}\mathbf{x} = \mathbf{b}$ has a solution, amounts to whether \mathbf{b} is in the span of the columns of \mathbf{A} (also known as the *column space* or *range*).

Consequently, for the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ to have a solution for all values of $\mathbf{b} \in \mathbb{R}^m$, it is required that the column space of \mathbf{A} is also apart of \mathbb{R}^m . This imposes our first condition, that $\mathbf{A} \in \mathbb{R}^n$ must have at least m columns; i.e. $n \geq m$. This is known as a necessary condition, but not a sufficient one. This is because, it is possible that some of the columns are redundant. Consider a 2x2 matrix where both columns are identical. As such, the matrix has the same column space as a 2x1 matrix, which does not span all of \mathbb{R}^2 , since the 2x1 is a real line (\mathbb{R}).

This redundancy is referred to as a *linear dependence*. Specifically, a set of vectors is *linearly independent* if there exists no vector in the set $\{\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(m)}\}$ that can be constructed from the other vectors. For example, if we add a vector to the set which is a combination of those already apart of it, then the new vector does not add any new points to the span. This therefore requires the column space of $\mathbf{A} \in \mathbb{R}^m$ to contain at least m linearly independent columns; this defines both the *necessary and sufficient conditions* for a linear system.

Therefore, we are ready to define the inverse solution to the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. We re-iterate that the vector space of the dependent variable \mathbf{x} occupies a \mathbb{R}^m vector space. The column space of $\mathbf{A} \in \mathbb{R}^n$ must contain at least $n \geq m$ columns, of which there are m -linearly independent vectors. Since the solution of the system must have only 1 solution per value of \mathbf{b} , we are therefore left with the requirement that $n = m$. Otherwise we could construct a number of solutions via a set of parameterised vector spaces.

All together, this means that our \mathbf{A} matrix must be *square* and all columns are *linearly independent*. Such a matrix is referred to as *non-singular*.

1.5 Norms

Within the context of vector spaces, the size of a vector is given by its *norm*, defined by three key properties:

1. Positive Definiteness: $f(\mathbf{x}) = 0$ for $\mathbf{x} = 0$.
2. Absolute Homogeneity: $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x}) \forall \alpha \in \mathbb{R}$.
3. Triangular Inequality: $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$.

They are functions that map vectors to non-negative scalar values, and are often interpreted as a distance between two points in the vector space. A common definition is the L^p norm given by:

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{1/p}, \quad (25)$$

where $p \in \mathbb{R}$, and for $p > 1$. For $p = 2$, L^2 is known as the *Euclidean norm*, which is often denoted as $\|\mathbf{x}\|$.

Similarly, the size of a matrix \mathbf{A} , can be obtained by the *Frobenius norm*:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}, \quad (26)$$

which is analogous to the L^2 norm of a vector.

1.6 Trace Operator

The trace operator is defined as the sum of all diagonal elements of a matrix:

$$\text{Tr}(\mathbf{A}) = \sum_i \mathbf{A}_{i,i}. \quad (27)$$

Note 5

If \mathbf{A} is not square (singular) a solution is still possible. However, matrix inversion is not a viable method for solving the system. For example one would need to consider dimensionality reduction methods, or analysing the rank/null space of the matrix; e.g.:

- Pseudoinverse
- Singular Value Decomposition (SVD)
- Iterative methods

These approaches however are costly from a computational perspective.

The trace is particularly useful due to specific properties that can be leveraged to simplify calculations. For example, the inner Frobenius norm of a matrix can be defined using:

$$||\mathbf{A}_F|| = \sqrt{\text{Tr}(\mathbf{A}\mathbf{A}^T)}. \quad (28)$$

It is also useful for manipulating linear system expressions via identities, such as:

- **Transpose Invariance:** The transpose of a matrix is invariant under the trace:

$$\text{Tr}(\mathbf{A}) = \text{Tr}(\mathbf{A}^T). \quad (29)$$

- **Cyclic Permutation:** A matrix comprised of multiple matrix factors is cyclically invariant:

$$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA}) \quad (30)$$

$$\rightarrow \text{Tr} \left(\prod_i^n \mathbf{A}^{(i)} \right) = \text{Tr} \left(\mathbf{A}^{(n)} \prod_i^{n-1} \mathbf{A}^{(i)} \right). \quad (31)$$

where this property holds even if two matrices have different shapes, e.g. $\mathbf{A}^{(1)} \in \mathbb{R}^{n \times m}$ and $\mathbf{A}^{(2)} \in \mathbb{R}^{m \times n}$.

1.7 Diagonal Matrix

Diagonal matrices, as per the name, contain non-zero elements in only the diagonal entries of the matrix:

$$\mathbf{D}_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j, \end{cases} \quad (32)$$

where one example has already been presented; the identity matrix \mathbf{I} . It is often the case that a square diagonal matrix is denoted as $\text{diag}(\mathbf{v})$, since the diagonal elements of the matrix are given by the vector \mathbf{v} . They are particularly useful since a diagonal matrix is computation efficient, since one can perform the operation $\text{diag}(\mathbf{v})\mathbf{x}$, by multiplying each element x_i by v_i :

$$\text{diag}(\mathbf{v})\mathbf{x} = \mathbf{v} \odot \mathbf{x}. \quad (33)$$

Of course non-square matrices that are diagonal are also valid, they are typically denoted however as normal matrices, and can still perform matrix-vector operation as per normal $\mathbf{D}\mathbf{x}$.

Note 6

We have made use of the Hadamard operation here, see above.

1.8 Symmetric Matrix

A symmetric matrix is equal to its transpose:

$$\mathbf{A} = \mathbf{A}^T. \quad (34)$$

For the topic of machine learning, they often arise from systems where there is a symmetry in the order of arguments, e.g. a distance measure between two points in a vector space. E.g. the $L^{(p)}$ norms are often symmetric on whether you are measuring the distance from $i \rightarrow j$, or $j \rightarrow i$; i.e. $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$.

1.9 Orthogonal Matrix

An orthogonal matrix is a square matrix in which the rows(columns) are mutually orthonormal:

$$\mathbf{A}^T \mathbf{A} = \mathbf{A}^{-1} \mathbf{A} = \mathbf{I}. \quad (35)$$

This naturally means that $\mathbf{T}^T = \mathbf{A}^{-1}$, which is advantageous to ML methods because such a system is cheap to compute the inverse of.

2 Probability & Information Theory

In this chapter an overview of probability and information theory will be provided.

Probability theory is a mathematical framework on which the idea of quantifying uncertain statements is based. Although there are several probabilistic interpretations, they utilise the same axiomatic principles to express mathematically the idea of a probability. The most common of these interpretations is that of Kolmogorov's axioms [2]. Whilst probability theory allows us to make uncertain statements, information theory forms the foundation on which we quantify the amount of uncertainty in a probability distribution. Together these two concepts formulate a framework in which machine learning/artificial intelligence learns to reason.

2.1 Random Variable

A random variable is a mathematical formulation of a quantity that depends on random events. It is denoted as a capital roman type faced letter, such as X , that is a measurable function from a sample space Ω to an event space E ; $X : \Omega \rightarrow E$. The sample space Ω denotes the space of all possible states, and the event space E is the outcome of each state. For example, in a coin flip, the sample space is the set $\{\text{Heads}, \text{Tails}\}$, and the event space is assigned values $\{+1, -1\}$. To denote the values that an event take on from a random variable, one uses the notation:

$$x \sim X, \quad (36)$$

where the lower case x is a value of the random variable (e.g. ± 1). The question you should now be asking is what defines the distribution of values that x can take?

2.2 Discrete Variables & Probability Mass Functions

The *probability distribution*, over discrete variables is described the concept of a *probability mass function* (PMF). Probability mass functions are denoted by a roman capital P . The PMF maps from the state in sample space Ω to the probability of the outcome x in the event space E :

$$P(X = x) = P(\{w \in \Omega | X(w) \in S\}), \quad (37)$$

where $w \subseteq \Omega$ is a subset of states in the sample space that give rise to the subset of outcomes $S \subseteq E$ in the event space.

The PMF conforms to what is known as the Kolmogorov axioms of probability theory. Specifically:

Axiom 1 - Positivity : $P(E) \in \mathbb{R}, P(E) > 0 \forall E \in F$

Axiom 2 - Unit Measure : $P(\Omega) = 1$

Axiom 3 - σ -additivity : $P(\bigcup_i E_i) = \sum_i P(E_i)$,

where we can now introduce the σ -algebra F in order to formally define the probability triple (Ω, F, P) .

For example, consider a single discrete random variable X , with k different possible states. The probability of selecting $x_k \sim X$ is uniformly equivalent, meaning that the PMF is:

$$P(X = x_i) = \frac{1}{k}, \quad (38)$$

for all possible i states. This meets the requirements of a PMF, because:

$$\sum_i^k P(X = x_i) = \sum_i^k \frac{1}{k} = 1, \quad (39)$$

is unit normalised.

2.3 Continuous Variables & Probability Density Functions

In the case of continuous variables, the probabilistic distribution of random variable values is dictated by a *probability density function* (PDF); denoted by lower case roman $p(x)$. Similarly to the discrete case, the Kolmogorov axioms apply, with the notational difference only of:

Axiom 1 - Positivity : $\forall x \ p(x) \geq 0$

Axiom 2 - Unit Measure : $\int p(x)dx = 1$.

It should be carefully noted that the PDF $p(x)$ does not correspond to the state, rather it provides the probability of landing in the infinitesimal region with volume δx . With this in mind, the example of a uniformly discrete set of states in Section 2.2, can be cast into the continuous limit, by asking what the probability the x lies in some set S . Where for a uniform distribution the set lies in the interval range $S \in [a, b]$. Therefore:

$$P(X = x) = \int_a^b p(x)dx, \quad (40)$$

where the PDF of a uniform distribution is $p(x; a, b) = \frac{1}{b-a}$. We have used the conditional notation of ';' to denote that the function $p(x)$ has some configurable parameters (a,b) that define the shape and characteristics of the distribution.

2.4 Marginal Probability

It is often the case that the system is defined by a set of random variables (X, Y) , with a *joint probability distribution* given by $P(X, Y)$. The probability of obtaining a specific value of x , irrespective of the value y , i.e. the probability distribution over a subset of the dependent variables, is known as the *marginal probability distribution*. This is more formally stated as:

$$\forall x \in X, P(X = x) = \sum_y P(X = x, Y = y), \quad (41)$$

which is known as the sum rule. For continuous variables, the sum is replaced by the integral:

$$p(x) = \int p(x, y) dy. \quad (42)$$

Note 7

The name *marginal* probability comes from the process of computing the marginal probabilities on paper. Specifically, if the values $P(X, Y)$ are placed in a grid with x -values in columns, and y -values in rows, then one writes $P(X = x)$ to the right of each row (in the margin) by summing across the columns.

2.5 Conditional Probability

It is also often the case that we are interested in the probability of an event with some value $X = x$, given a second random variable $Y = y$. This is known as a *conditional probability*, which is formulated as:

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(X = x)}, \quad (43)$$

where $P(X = x|Y = y)$ is the conditional probability. It should be noted that a conditional probability is only defined for $P(X = x) > 0$, since a conditional probability of an event that never happened is impossible.

2.6 Chain Rule of Conditional Probabilities

The joint probability over n random variables, denoted as $x^{(i)}$ for $i \in [1, n]$, can be decomposed into the product of conditional probabilities over a single variable:

$$P(x^{(1)}, \dots, x^{(i)}) = \prod_k^n P(x^{(k)} | \bigcap_{i=1}^{k-1} x^{(i)}). \quad (44)$$

This is known as the *chain rule*, or *product rule*, of probability. For simplicity, and as an illustrative example, we consider the simple case of three random variables a, b, c :

$$P(a, b, c) = P(a|b, c)P(b, c) = P(a|b, c)P(b|c)P(c). \quad (45)$$

2.7 Independence and Conditional Independence

Two random variables X and Y , are *independent* if the probability distributions can be expressed as the product of two marginal distributions:

$$\forall x \in X, y \in Y, p(X = x, Y = y) = p(X = x)p(Y = y). \quad (46)$$

Similarly, for two random variables X , and Y conditionally independent on a third random variable Z , then the conditional independence can be factorised over the conditional variable Z :

$$\forall x \in X, y \in Y, z \in Z, p(X = x, Y = y | Z = z) = p(X = x | Z = z)p(Y = y | Z = z). \quad (47)$$

In both cases, a short hand exists to denote the independence:

$$p(X = x, Y = y) = p(X = x)p(Y = y) : \rightarrow x \perp y \quad (48)$$

$$p(X = x, Y = y | Z = z) = p(X = x | Z = z)p(Y = y | Z = z) : \rightarrow x \perp y | z. \quad (49)$$

2.8 Expectation, Variance, and Covariance

The *expectation*, or *expected value*, of a function $f(x)$ with respect to a probability distribution $P(X)$, is given by the mean value of the function at each value of x X drawn from $P(x)$. For discrete variables, this can be computed by the summation:

$$\mathbb{E}_{x \sim P}[f(x)] = \sum_x P(x)f(x), \quad (50)$$

while for the continuous variables the sum is replaced by the integral formulism:

$$\mathbb{E}_{x \sim P}[f(x)] = \int_x p(x)f(x)dx. \quad (51)$$

Of course the expectation applies to the joint and conditional probabilities defined in the previous sections. For completeness these are defined below for the joint distribution $p(x, y)$:

$$\mathbb{E}_{X,Y}[f(X, Y)] = \int \int f(x, y)p(x, y)dxdy,$$

conditional probability:

$$\mathbb{E}_{X|Y=y}[f(X, Y)] = \mathbb{E}[f(X, Y)|Y = y] = \int f(x, y)p(x|y)dx.$$

It should be noted that the expectation is linear:

$$\mathbb{E}_X[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_X[f(x)] + \beta \mathbb{E}_X[g(x)], \quad (53)$$

where α , and β are scalars and independent of x .

The *variance* of a random variable is given by:

$$\text{Var}(f(x)) = \mathbb{E}_X \left[(f(x) - \mathbb{E}_X[f(x)])^2 \right], \quad (54)$$

such that the *standard deviation* of the random variable, i.e. the error associated with the varying nature of X , is given by $\sigma_X = \sqrt{\text{Var}(\cdot)}$.

The *covariance* of two random variables X and Y , or a measure of how linearly related the two variables are, is given by:

$$\text{Cov}(f(x), g(x)) = \mathbb{E}_{X,Y}[(f(x) - \mathbb{E}_X[f(x)])(g(y) - \mathbb{E}_Y[g(y)])], \quad (55)$$

where for large positive values of covariance the two variables both deviate from the mean together, and for negative large values, they deviate together away from their respective means, but in opposite directions.

Note 8

The expectation notation can often be abbreviated when the context of the distribution is clear. For example:

$$\mathbb{E}_{x \sim p(x)}[f(x)] \rightarrow \mathbb{E}_X[f(x)]. \quad (52)$$

This make use of the fact the only dependent random variable is X .

2.9 Key Probabilistic Distributions

2.9.1 Gaussian Distribution

The *Gaussian*, or *normal distribution*, is likely the most common used distribution in machine learning, statistics, and information theory. This is somewhat in part to the *central limit theorem*, which states that for a collection of n random variables, $\{X_1, \dots, X_n\}$, the sum of the random variables will asymptotically approach a Gaussian distribution as $n \rightarrow \infty$.

The Gaussian distribution is given by:

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu)^2\right), \quad (56)$$

where the two parameters $\mu \in \mathbb{R}$, and $\sigma \in (0, \infty)$, are the mean and standard deviation of the distribution. In multiple dimensions (\mathbb{R}^d), the Gaussian distribution generalises well to the multivariate Gaussian/Normal distribution:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{\frac{1}{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2} ((\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}))^2\right), \quad (57)$$

where $\boldsymbol{\mu}$ is a vector valued object giving the mean of the distribution for each dimension of \mathbf{x} , and $\boldsymbol{\Sigma}$ is the covariance matrix of the distribution with shape $\mathbb{R}^{d \times d}$.

2.9.2 Dirac and Empirical Disitribution

The *Dirac* distribution is defined as:

$$p(x) = \delta(x - \mu) = \begin{cases} \infty, & \text{if } x = \mu \\ 0, & \text{if } x \neq \mu, \end{cases} \quad (58)$$

meaning that it is zero in the entire domain of x except when identically equivalent to μ . It is a useful distribution, because it assigns all probability mass to a singular point in the x space. The Dirac function is not an ordinary function, that associates with each x a real valued output; i.e. a real valued domain to codomain map $f : X \rightarrow \mathbb{R}$. It is instead known mathematically as a *generalisation* function, that is defined in terms of its properties when integrated. Specifically:

$$\int_{-\infty}^{\infty} \delta(x) dx = 1, \quad (59)$$

with a few key properties:

Note 9

It should be carefully noted that the concept of covariance and dependence are related, but distinct concepts. Specifically:

- **Independent Variables**
 $\Rightarrow \text{Cov}(f(x), f(y)) = 0$
- $\text{Cov}(f(x), f(y)) \neq 0 \Rightarrow$ **Dependent Variables**,

where we have been careful to use a directional arrow. This is because *independence* is distinct from covariance and more strict. This is because for two variables to have zero covariance, there must be no linear dependence between them. Independence on the other hand excludes non-linear relationships as well.

- **Scaling/Symmetry:** $\int_{-\infty}^{\infty} \delta(\alpha x) dx = \int_{-\infty}^{\infty} \delta(\mu) \frac{d\mu}{|\alpha|} = \frac{1}{|\alpha|}$
- **Distributive product** $(x - a)^n \delta(x - a) = 0$
- **Linearity:** $\forall x, xf(x) = xg(x)$, where f and g are distributions, then $f(x) = g(x) + c\delta(x)$ for a constant c
- **Translation:** $\int_{-\infty}^{\infty} f(x)\delta(x - T)dx = f(T)$

It has significant use in machine learning applications, because it defines the *empirical* distribution for a finite sample of a random variable X with m entries:

$$\hat{p}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^m \delta(\mathbf{x} - \mathbf{x}_i), \quad (60)$$

which put mass $1/m$ on each of the m points in the dataset $\{x_1, \dots, x_m\}$.

2.10 Bayes Rule

Bayes Rule provides a mathematical rule for inverting the conditionality of probabilities, e.g. $P(X|Y)$ and $P(Y|X)$:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}. \quad (61)$$

It can be derived from the definition of conditional probability. Specifically:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad (62)$$

where $P(A \cap B)$ represents the probability that both A and B are true. Consequently, it is similarly true that:

$$P(B|A) = \frac{P(B \cap A)}{P(A)}. \quad (63)$$

Since $P(A \cap B) = P(B \cap A)$, one obtains Bayes theorem.

2.11 Measure Theory & Continuous Variables

Measure theory is the mathematical study of geometrical measures (e.g. length, area, and volume) in addition to the concept of mass, magnitude and probability in a vector space. Whilst out of the scope of this course, it is important to understand one concept of measure theory for continuous variables. Consider, two random variables X and Y , that are deterministic functions of each other: $y = g(x)$. In this instance g is a continuous, invertible, and differentiable transformation. As such, it is natural to think that:

$$p_y(y) = p_x(g^{-1}(y)). \quad (64)$$

Note 10

Bayes rule is named after Reverend Thomas Bayes, who first proposed the rule for the special case of the binomial distribution. However, it was Laplace who presented the generic form we known today, after independently deriving it. Both Richard Price and Pierre-Simon Laplace published recognition of Thomas Bayes contributions posthumously.

However, this is incorrect. To understand this we will see a simple example, and then a more concrete generalisation that links to the idea of measure theory. For the simple example, consider two scalar random variables X and Y that are related via $y = x/2$, and $x \sim \mathcal{U}(0, 1)$, i.e. we sample from a uniform distribution for x . Under the rule $p_y(y) = p_x(2y)$, we have:

$$p_y(y) = \begin{cases} 0, & \text{if } x \notin [0, 1/2] \\ 1, & \text{if } x \in [0, 1/2]. \end{cases} \quad (65)$$

Consequently, the integral of the distribution is given by:

$$\int_0^1 p_y(y) dy = \int_0^{1/2} \frac{1}{b-a} dx = \frac{1}{2}, \quad (66)$$

where $b = 1, a = 0$ are constants of the uniform distribution \mathcal{U} . It should be immediately obvious that this violates the second axiom of unit measure for probability theory.

This problem originates from a failure to account for the distortion of the domain space of x by the function g . If you recall from sub-section 2.3, the probability density function for x , corresponds to the probability of the value residing in the infinitesimal volume of δx . Since the function $g : X \rightarrow Y$ can expand or contract the space, the infinitesimal volume in δy may be larger or smaller. Consequently, to solve this problem, we must start with conserving the correct quantity; specifically:

$$|p_y(g(x)) dy| = |p_x(x) dx|. \quad (67)$$

Solving this, we obtain:

$$p_y(y) = p_x(g^{-1}(y)) \left| \frac{dx}{dy} \right|, \quad (68)$$

or equivalently:

$$p_x(x) = p_y(g(x)) \left| \frac{dg(x)}{dx} \right|. \quad (69)$$

In higher dimensions, the derivative generalises to the determinant of the *Jacobian matrix*, where $J_{i,j} = \frac{dx_i}{dy_j}$, and $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ vector spaces:

$$p_x(\mathbf{x}) = p_y(g(\mathbf{x})) \left| \det \left(\frac{dg(\mathbf{x})}{d\mathbf{x}} \right) \right|. \quad (70)$$

2.12 Information Theory

Information theory, a branch of applied mathematics, is concerned with quantifying the amount of information present in a signal. Its origins stem from the problem of sending messages using discrete alphabets using a noisy communication channel. For example, radio transmissions using morse code. Therefore, in this context information theory is useful in constructing optimal codes, and the length of a message when sampled from a probability distribution. This section is not a complete review of information theory, but we concentrate on the characterisation of probability distributions, and/or to quantify the similarity between probability distributions.

The most basic intuition of information theory is the realisation that an unlikely event provides more information than a likely event. In this regard, the unlikely event is more *informative*, and thus of more use. Therefore, we want to formulate information within this context. To achieve this we define three key criteria:

- Likely events have little information content
- Unlikely events have a lot of information content
- Independent events add information.

To satisfy these three requirements, we define the *self-information* of an event $X = x$:

$$I(x) = -\ln(P(x)). \quad (71)$$

This definition of the information can be used to quantify the amount of information in an entire distribution, using the ensemble of events m , via *Shannon Entropy*:

$$H(X) = \mathbb{E}_{x \sim P}[I(x)] = -\mathbb{E}_{x \sim P}[\ln(P(x))]. \quad (72)$$

The intuitive interpretation of Shannon entropy, is that the entropy of the distribution P is equivalent to the expectation of surprise. This is because $-\ln(P) = \ln(1/P)$, therefore if $P(x) \rightarrow 1$, then the event offers little surprise, and so the entropy is close to 0. On the other hand, for low $P(x)$ the unlikely event has a lot of surprise, and with it provides a lot of information, or entropy. This interpretation gives a lower bound on the number of bits/nats, needed to on average encode symbols drawn from the distribution P .

With this defined, we are now able to define one of the most significant concepts and useful concepts in machine learning. The idea of a *distance measure*, between two distributions P and Q , over the same random variable X :

$$D_{\text{KL}}(P||Q) = \mathbb{E}_{x \sim P} \left[\ln \left(\frac{P(x)}{Q(x)} \right) \right] = \mathbb{E}_{x \sim P} [\ln(P(x)) - \ln(Q(x))], \quad (73)$$

which is the difference in entropy between the two distributions, known as the *Kullback-Leibler divergence* (KL-div). It should be noted that the KL-div is not a true metric. This is because for a distance measure to be a metric, it must adhere to four key properties:

- The distance to itself is 0: $d(P, P) = 0$
- The distance is always positive (positivity): $d(P, Q) > 0$ for $P \neq Q$
- The distance between the two distributions is symmetric or irrespective of order of calculation: $d(P, Q) = d(Q, P)$
- The triangular inequality holds: $d(P, Z) \leq d(P, Q) + d(Q, Z)$, where Z is a third distribution or point in the vector space.

Since the KL-div does not adhere to the symmetric property it is not a true metric. As such, there can be drastic consequences of using $D_{\text{KL}}(P||Q)$ or $D_{\text{KL}}(Q||P)$.

Note 11

The information stored in $I(x)$ depends on the logarithm that is used. If it is base e then the unit of information is a *nats*, which is the amount of information stored in giving an event probability $1/e$. If the base is 2, then it is a *bit*. All these units are just rescalings of the same concept.

Another important quantity that is closely related to the KL divergence, is the *cross-entropy*:

$$H(P, Q) = H(P) + D_{\text{KL}}(P||Q) \quad (74)$$

$$= -\mathbb{E}_{x \sim P}[\ln(P(x))] + \mathbb{E}_{x \sim P}[\ln(P(x)) - \ln(Q(X))] \quad (75)$$

$$= -\mathbb{E}_{x \sim P}[\ln(Q(x))] = \sum_i P(x) \ln(Q(x)) \quad (76)$$

It will be shown later how the the Kullback-Leibler divergence, cross-entropy, and negative log-likelihood, three fundamentally important measure of similarity are equivalent when training a machine learning model, i.e. an optimisation problem.

Note 12

From a practical perspective in machine learning, you will encounter expressions with $\ln(0)$. We often *clip* the function to 0 due to the convergence rate of $\lim_{x \rightarrow 0} x \ln(x) = 0$.

3 ML 101

At its core, machine learning is an algorithm that can learn from data patterns, in order yield some actionable information. There are many famous quotes about the overarching focus of machine learning. In my opinion, Mitchell (1997) [3], states it best:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

In this setup there are three key components to a machine learning algorithm:

Experience 'E' : The stimulus that drives learning, is the set of examples $x \in \mathbb{R}^n$, or a dataset \mathcal{D} of many examples $x = x_m$

Task 'T' : Given an example x the model f should learn a prediction $y = f(x)$

Performance 'P' : Evaluate the performance of the model f , often via the foundations of *measure theory*¹; generalisation of geometric distances for a measureable space (Ω, F) , such that for a measure $\mathbb{M} : F \rightarrow [0, \infty]$.

A brief description is given below for each of these concepts.

Experience A machine learning algorithm *experiences* a dataset of m data points, each a realisation of a random variable/probability distribution $x^{(i)} \sim X^{(i)}$. There are two predominant types of datasets, that in turn define the learning algorithm:

$$\mathcal{D} \in \begin{cases} \{(x^{(i)}, y^{(i)}), \in \mathbb{R}^n \times \mathbb{R}^l\}, & : \text{ supervised} \\ \{(x^{(i)}), \in \mathbb{R}^n\}, & : \text{ unsupervised,} \end{cases} \quad (77)$$

¹Said that this would come back later in the notes, see Section 2.12

where the new additional variable y is a label/target. In the former case, the dataset encourages a *supervised learning* algorithm, whilst the latter an *unsupervised learning* algorithm. In a supervised learning algorithm, the task is one of classifying each data point according to its label/target. In unsupervised learning, the goal is to learn useful properties of the distribution, whether it be explicitly by density estimation, or implicitly via denoising or clustering.

Note 13

It is however at this point that a **fundamental** concept must be understood. Unsupervised learning attempts to learn by data point observations of a random vector \mathbf{x} , the density $p(\mathbf{x})$. In supervised learning the algorithm attempts to learn via paired data (\mathbf{x}_i, y_i) to predict the label/target y , where in doing so the algorithm is learning $p(y|\mathbf{x})$.

However, formally the two types of algorithms are not defined, with the boundary often blurred between them. To illustrate this, consider a simple dataset $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^n\}_m$, in which the task is to learn the density $p(\mathbf{x})$. Using the chain rule for probability states:

$$p(\mathbf{x}) = \prod_i^n p(x_i | x_1, \dots, x_{i-1}). \quad (78)$$

As a result of this decomposition, the task of learning the density is broken down into n conditional density estimation problems. As such, whilst the dataset originally encouraged a unsupervised algorithm, it is technically now indistinguishable from a supervised learning algorithm.

Conversely, we can consider the opposite problem, in which we are given the dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^l\}_m$. The challenge now is to learn the conditional label probability of $p(y|\mathbf{x})$. Using the definition of conditional probability (see equation 43):

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_d p(\mathbf{x}, y')}, \quad (79)$$

meaning that one can employ an unsupervised learning density estimation method to learn the joint distribution $p(\mathbf{x}, y)$, then infer the label/target conditioned probability.

Performance The performance of an algorithm is key to determining the optimised solution obtained during training. The performance measures used are dependent on the problem, and the algorithm. For example, in a supervised learning classification problem, one might consider the *accuracy* with which one correctly predicts the label $\text{acc.} = N_y/N$, where N_y is the number of correctly labeled data points, and N the total number of data points in the dataset.

Task The task of the machine learning algorithm can be wide spread with some examples given below:

- Classification

- Regression
- Transcription
- Anomaly detection
- Synthesis/Generative
- Denoising
- Density Estimation

3.1 Estimators

A point estimator is any function that takes as an argument a dataset of m independently identically distributed data points, that infers from the ensemble of data a quantity of interest. This quantity of interest, denoted as θ , could be the parameters of a linear equation, or the lifetime of a nuclear decaying element. To denote the estimated parameters from the true values, the *hat* notation is used, $\hat{\theta}$:

$$\hat{\theta} = g(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}). \quad (80)$$

The function $g(\cdot)$ is not restricted to being of a functional form that must return a value that is close to the true value of θ , nor its range. Whilst this offers substantial flexibility, the quality of the function is defined by its ability to accurately estimate the true value of θ . This quality is reflected in the concept of a point estimators bias and variance properties.

3.1.1 Bias

The bias of an estimator is defined as:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}_m) - \theta, \quad (81)$$

where the expectation is over the iid dataset, or random variable from which the samples are drawn. An estimator $\hat{\theta}_m$ is unbiased if $\text{bias}(\hat{\theta}_m) = 0$, which arises when $\mathbb{E}(\hat{\theta}_m) = \theta$. More formally, an estimator $\hat{\theta}_m$ is asymptotically unbiased if $\lim_{m \rightarrow \infty} \text{bias}(\hat{\theta}_m) = 0$, and so $\lim_{m \rightarrow \infty} \mathbb{E}(\hat{\theta}_m) = \theta$.

3.1.2 Example: Gaussian Estimators

Consider a set of samples $\mathcal{D} = \{x_1, \dots, x_m\}$ that are iid, and distributed according to a Gaussian distribution:

$$p(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(x_i - \mu)^2}{\sigma^2}\right). \quad (82)$$

To estimate the mean, we construct the function:

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^m x_i. \quad (83)$$

To determine the bias of this function, we estimate:

$$\text{bias}(\hat{\mu}) = \mathbb{E}[\hat{\mu}] - \mu \quad (84)$$

$$= \mathbb{E}\left[\frac{1}{m} \sum_i^m x_i\right] - \mu \quad (85)$$

$$= \left(\frac{1}{m} \sum_i^m \mathbb{E}[x_i]\right) \quad (86)$$

$$= \left(\frac{1}{m} \sum_i^m \mu\right) - \mu \quad (87)$$

$$= \mu - \mu = 0. \quad (88)$$

Therefore, the function we selected, known as the *sample mean*, is an unbiased estimator of the Gaussian mean.

However, let's now look at an estimator of the variance σ^2 . We construct first an analogous estimator to that of the mean, known as the *sample variance*:

$$\hat{\sigma}^2 = \frac{1}{m} \sum_i^m (x_i - \hat{\mu})^2, \quad (89)$$

where we have used the sample mean from above $\hat{\mu}$. The bias of this estimator is given by:

$$\text{bias}(\hat{\sigma}^2) = \mathbb{E}[\hat{\sigma}^2] - \sigma^2 \quad (90)$$

$$= \mathbb{E}\left[\frac{1}{m} \sum_i^m (x_i - \hat{\mu})^2\right] - \sigma^2 \quad (91)$$

$$= \frac{1}{m} \mathbb{E}\left[\sum_i^m ((x_i - \mu) - (\hat{\mu} - \mu))^2\right] - \sigma^2 \quad (92)$$

$$= \frac{1}{m} \mathbb{E}\left[\sum_i^m (x_i - \mu)^2 - 2(\hat{\mu} - \mu) \sum_i^m (x_i - \mu) + \sum_i^m (\hat{\mu} - \mu)^2\right] - \sigma^2 \quad (93)$$

$$= \frac{1}{m} \mathbb{E}\left[\sum_i^m (x_i - \mu)^2 - \frac{2}{m}(\hat{\mu} - \mu)(\hat{\mu} - \mu) + m(\hat{\mu} - \mu)^2\right] - \sigma^2 \quad (94)$$

$$= \left[\frac{1}{m} \mathbb{E}\left[\sum_i^m (x_i - \mu)^2\right] - \mathbb{E}[(\hat{\mu} - \mu)^2]\right] - \sigma^2 \quad (95)$$

$$= \left[\sigma^2 - \frac{1}{m}\sigma^2\right] - \sigma^2 = \frac{m-1}{m}\sigma^2 - \sigma^2 = -\frac{\sigma^2}{m}, \quad (96)$$

which is biased. As such, a simple adaptation of the function can be made that yields an unbiased estimator:

$$\hat{\sigma}^2 = \frac{1}{m} \sum_i^m (x_i - \hat{\mu})^2, \rightarrow \hat{\sigma}^2 = \frac{1}{\textcolor{red}{m}-1} \sum_i^m (x_i - \hat{\mu})^2, \quad (97)$$

which is outlined in red. This simple adaptation results in the unbiased estimator:

$$\mathbb{E}[\hat{\sigma}^2] = \mathbb{E}\left[\frac{1}{m-1} \sum_i^m (x_i - \hat{\mu})^2\right] \quad (98)$$

$$= \frac{m}{m-1} \left(\frac{m-1}{m} \sigma^2\right) = \sigma^2, \quad (99)$$

therefore the bias is $= \sigma^2 - \sigma^2 = 0$, i.e. a perfect estimator.

3.1.3 Variance & Standard Error

The second property that defines the quality of a point estimator, is the degree with which the estimated parameter $\hat{\theta}$ varies as a function of the dataset. The variance, defined as:

$$\text{Var}(\hat{\theta}_m) = \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2], \quad (100)$$

where one can also define the standard deviation/error, denoted as $\text{SE}(\hat{\theta})$, as the square root of the variance.

3.1.4 Example: Gaussian Estimators

As an example, we return to the Gaussian example of sub-section 3.1.2, in which we have a iid dataset from a Gaussian pdf, yielding the estimator:

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^m x_i. \quad (101)$$

We now want to quantify the variance of the estimator $\hat{\mu}$. Specifically:

$$\text{Var}(\hat{\mu}) = \text{Var}\left(\frac{1}{m} \sum_i^m x_i\right) \quad (102)$$

$$= \mathbb{E}\left[\frac{1}{m} \left(\sum_i^m x_i - \mathbb{E}\left(\sum_i^m x_i\right)\right)\right] \quad (103)$$

$$= \frac{1}{m^2} \sum_i^m \text{Var}(x_i) = \frac{\sigma^2}{m^2}. \quad (104)$$

This has the the desirable property that as the sample size increases, the variance decreases, meaning that our estimator approaches the correct value of the model that drives the data.

3.1.5 Bias and Variance Trade-off

The concepts of bias and variance for point estimators is one of a few key concepts to understand in machine learning, due to its simple but significant contributions to the behaviour of a ML algorithm

during training. Bias measures the expected deviation from the true value of the function or parameter. Variance provides a measure of the deviation from the expected estimator value, when repeating the experiment with a number of *iid* datasets.

As such, we are often faced with the choice of selecting an estimator with more bias or more variance. How do we navigate this decision? The question can be found in a mathematical certainty, imposed by considering the *mean squared error* of the estimates:

$$\text{MSE} = \mathbb{E} \left[(\hat{\theta} - \theta)^2 \right] \quad (105)$$

$$= \mathbb{E} \left[\hat{\theta}^2 - 2\hat{\theta}\theta + \theta^2 \right] \quad (106)$$

$$= \mathbb{E}[\hat{\theta}^2] - 2\mathbb{E}[\hat{\theta}]\theta + \theta^2 \quad (107)$$

$$= \text{Var}(\hat{\theta}) + \mathbb{E} \left[\hat{\theta} \right]^2 - 2\mathbb{E}[\hat{\theta}]\theta + \theta^2 \quad (108)$$

$$= \text{Var}(\hat{\theta}) + \text{bias}(\theta)^2. \quad (109)$$

The MSE we see here measures the difference between the estimator and the true value of the parameter θ . What it shows is that a desirable estimator must balance the bias and variance contributions. This is known as the bias-variance trade-off problem, in which the capacity of a ML model should scale with the complexity of the data. If the capacity exceeds the complexity you will be susceptible to a highly variance estimator. In contrast, a ML model that has a small capacity relative to the complexity of the data, will have insufficient degrees of freedom to express data behaviour, which means the estimator will be biased. This is demonstrated by Figure 1.

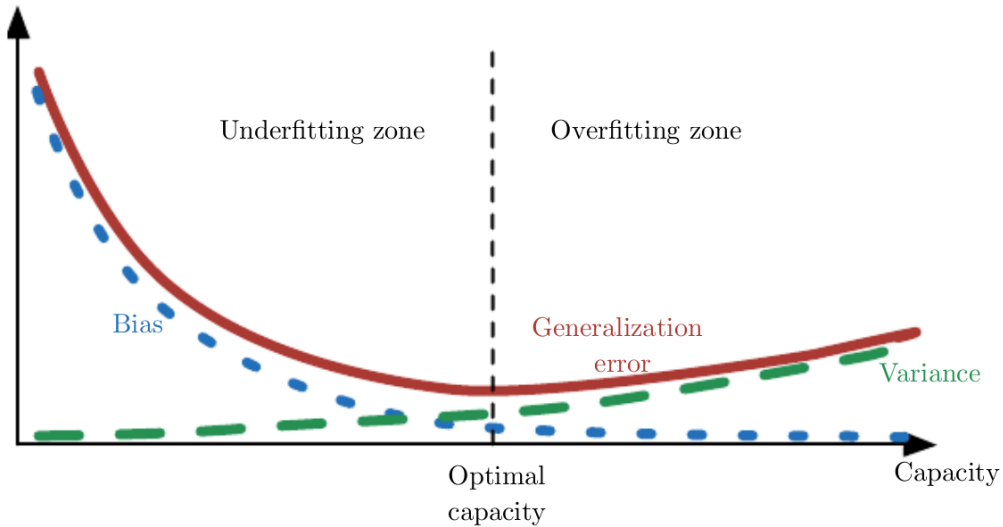


Figure 1: Bias-variance trade off problem, in which the x-axis shows the capacity of the ML model (e.g. size of a neural network).

3.2 Maximum Likelihood Method: Learning point estimator functions

We have implicitly defined a process of constructing point estimator functions, via the process of proposing a function, and then test said function for desirable properties such as bias/variance. This process however is cumbersome, and so we in principle want to define a process of deriving functions that are good estimators for a *family* of different models.

The most common strategy is the *maximum likelihood method* (MLM). Consider a dataset $\mathcal{D} = \{\mathbf{x}^{(i)} \in \mathbb{R}^n\}_m$ drawn from a distribution $p_{\text{data}}(\mathbf{x})$. This *true* distribution is unknown, and so must be learned. Therefore, we consider a parametric family of probability functions $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$, over the same space as \mathbf{x} , conditioned on some parameters $\boldsymbol{\theta}$. The goal is therefore to configure (learn) the parameters $\boldsymbol{\theta}$ such that the difference between p_{model} and p_{data} is zero. This can be achieved by the maximum likelihood estimator:

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} p_{\text{model}}(\mathcal{D}; \boldsymbol{\theta}) \quad (110)$$

$$= \arg \max_{\boldsymbol{\theta}} \prod_i^m p_{\text{model}}(\mathbf{x}^{(i)}; \boldsymbol{\theta}). \quad (111)$$

This form of the MLM is problematic for machine learning learning, due to the fact that a computer must represent a real value with a finite bit representation. Specifically, a computer has limited resources to represent a float-point value. As such, the product of many potentially small or large probabilities would result in numerical underflow/overflow. To prevent this, it is convenient to utilise the log-likelihood:

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} \sum_i^m \log \left(p_{\text{model}}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) \right). \quad (112)$$

Since the argument maximum operation ($\arg \max$) does not change when rescaling the function, we can divide by the log likelihood by the number of data points m in the dataset, which expresses the log-likelihood as a

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} \frac{1}{m} \sum_i^m \log \left(p_{\text{model}}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) \right) = \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log \left(p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) \right). \quad (113)$$

3.3 Conditional Log-Likelihood and Mean Squared Error

The maximum likelihood estimator can also be defined for conditional probability distributions, $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$. This is key, since it is primarily the problem set of supervised learning algorithms. In this instance, $\{\mathbf{x}^{(i)}\}_m$ represents all inputs, and $\{\mathbf{y}^{(i)}\}_m$ are the data labels/targets, such that the MLM is defined as:

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} P(\{\mathbf{y}^{(i)}\}_m | \{\mathbf{x}^{(i)}\}_m), \quad (114)$$

which for iid samples can be decomposed into:

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} \sum_i^m \log \left(P(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}) \right) \quad (115)$$

3.4 Kullback-Leibler \leftrightarrow Maximum Likelihood Method

Maximum likelihood methods are a powerful technique for estimating a probability density function from examples alone (a dataset). It is however not always obvious as to why the product of probabilities, when maximised for the model parameters, over the examples drawn from $p_{\text{data}}(\mathbf{x})$, forces the model probability ($p_{\text{model}}(\mathbf{x}|\boldsymbol{\theta})$) to be the same. One way to interpret this method is to consider the MLM as minimising the dissimilarity between the empirical distribution formed by the dataset ($\tilde{p}_{\text{data}}(\mathbf{x})$) and the model² as measured by the KL-div:

$$\mathcal{D}_{\text{KL}}(P_{\text{data}}||P_{\text{model}}) = \mathbb{E}_{x \sim P} [\ln(P_{\text{data}}) - \ln(P_{\text{model}})] . \quad (116)$$

Naturally, the parameters of the ML model $\boldsymbol{\theta}$ are optimised by minimising the above dissimilarity measure:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \underbrace{\int p_{\text{data}}(x) \log(p_{\text{data}}(x)) dx - \int p_{\text{data}}(x) \log(p_{\text{model}}(x|\boldsymbol{\theta})) dx}_{= 0 \text{ as } \text{no } \boldsymbol{\theta}} \quad (117)$$

$$= \arg \min_{\boldsymbol{\theta}} - \underbrace{\int p_{\text{data}}(x) \log(p_{\text{model}}(x|\boldsymbol{\theta})) dx}_{p_{\text{data}}(x) = \frac{1}{m} \sum_i^m \delta(x - x_i)} \quad (118)$$

$$= \arg \min_{\boldsymbol{\theta}} - \int \left[\frac{1}{m} \sum_i^m \delta(x - x_i) \right] \log(p_{\text{model}}(x|\boldsymbol{\theta})) dx \quad (119)$$

$$= \arg \min_{\boldsymbol{\theta}} - \frac{1}{m} \sum_i^m \log(p_{\text{model}}(x|\boldsymbol{\theta})) \quad (120)$$

$$= \arg \min_{\boldsymbol{\theta}} \mathbb{E}_{x \sim P} \log(p_{\text{model}}(x|\boldsymbol{\theta})) . \quad (121)$$

What this results demonstrates, is that by minimising the KL-div, or dissimilarity, the negative log-likelihood is maximised between the model and empirical distribution defined by the data. This is essentially a density matching problem, and intuitively means that the MLM is just a process of minimising a *distance measure* (see Section 2.12).

References

- [1] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20081110. Oct. 2008. URL: <http://www2.imm.dtu.dk/pubdb/p.php?3274>.
- [2] A.N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. 1. Springer Berlin, Heidelberg, 1933, pp. V, 62. DOI: <https://doi.org/10.1007/978-3-642-49888-6>.
- [3] Tom Michael Mitchell. *Machine Learning*. Ed. by Eric M. Munson. McGraw-Hill Series in Computer Science. Boston, MA: WCB/McGraw-Hill, 1997, p. 414. ISBN: 0-07-042807-7.

²This concept is illustrated in Figure 2, in which the empirical distribution $\tilde{p}_{\text{data}}(x)$ is represented by an ensemble of point estimates of the true $p_{\text{data}}(x)$, whilst the model is a continuous distribution ($p_{\text{model}}(x)$)

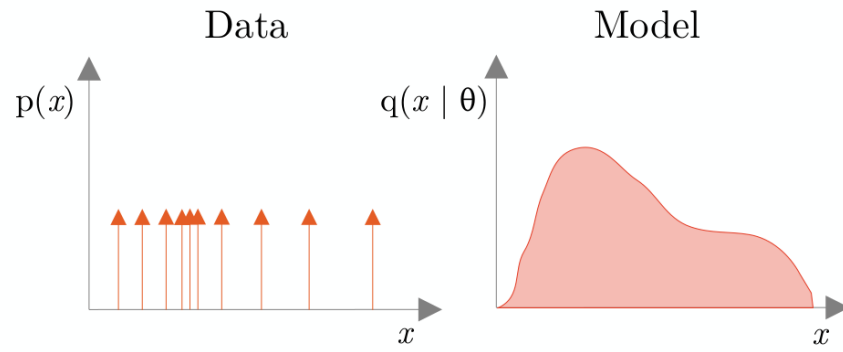


Figure 2: The figure on the left is an illustration of point estimates of a density function $p_{\text{data}}(x)$, that forms an empirical distribution. The right figure is the continuous distribution, produced by a ML model ($p_{\text{model}}(x)$).