Maximum Entropy Models: Summer Research Placement

Summer 2025

Ruby Keily-Thurstain

Supervised by Dr Marianna Cerasuolo

Abstract

This report explores the mathematical foundations of entropy in information theory, and its relationship to convex optimisation in the form of maximum entropy models, before applying these ideas to a real-world ecological modelling problem. We begin by developing the concept of entropy as a measure of uncertainty, and show how maximising entropy can be formulated as a convex optimisation problem. The properties of convexity are also discussed in detail, as they influence the construction and solutions of these optimisation problems.

To illustrate how this theory can be applied, we train a maximum entropy (MaxEnt) model on a species distribution problem: several Ariolimax (banana slug) species found along the Pacific coast of North America. Using temperature and precipitation data, we analyse the relationship between these environmental factors and the species distribution, and use the foundations of MaxEnt to predict the suitability of land parcels for the Ariolimax to inhabit. We conclude by discussing the strengths and limitations of the MaxEnt approach, and how it demonstrates the power of optimisation-based methods in ecological modelling.

Contents

1 Introduction

Entropy is a fundamental concept in both mathematics and physics, used to describe uncertainty, disorder and information. It can be utilised for mathematical modelling, by maximising the uncertainty in the predictions, giving us a principled way to make predictions and assign probabilities when only limited information is available. The mathematical foundations of this principle are deeply linked with convexity, as many maximum entropy problems can be framed as convex optimisation problems.

In this report, we first look at the theoretical background that connects entropy and optimisation. We will discuss the properties of convex functions and convex optimisation problems, and explore the well-behaved solutions these structures yield. We then move on to the idea of maximising entropy with various constraints, which underpins many modelling techniques, including the maximum entropy (MaxEnt) models used is species distribution modelling.

To put these ideas in context, we apply a MaxEnt model to the distribution of Ariolimax (banana slugs) across the Pacific coast of North America. Using occurrence and environmental data, we will explore how optimisation theory can be used to predict where a species is likely to live and thrive.

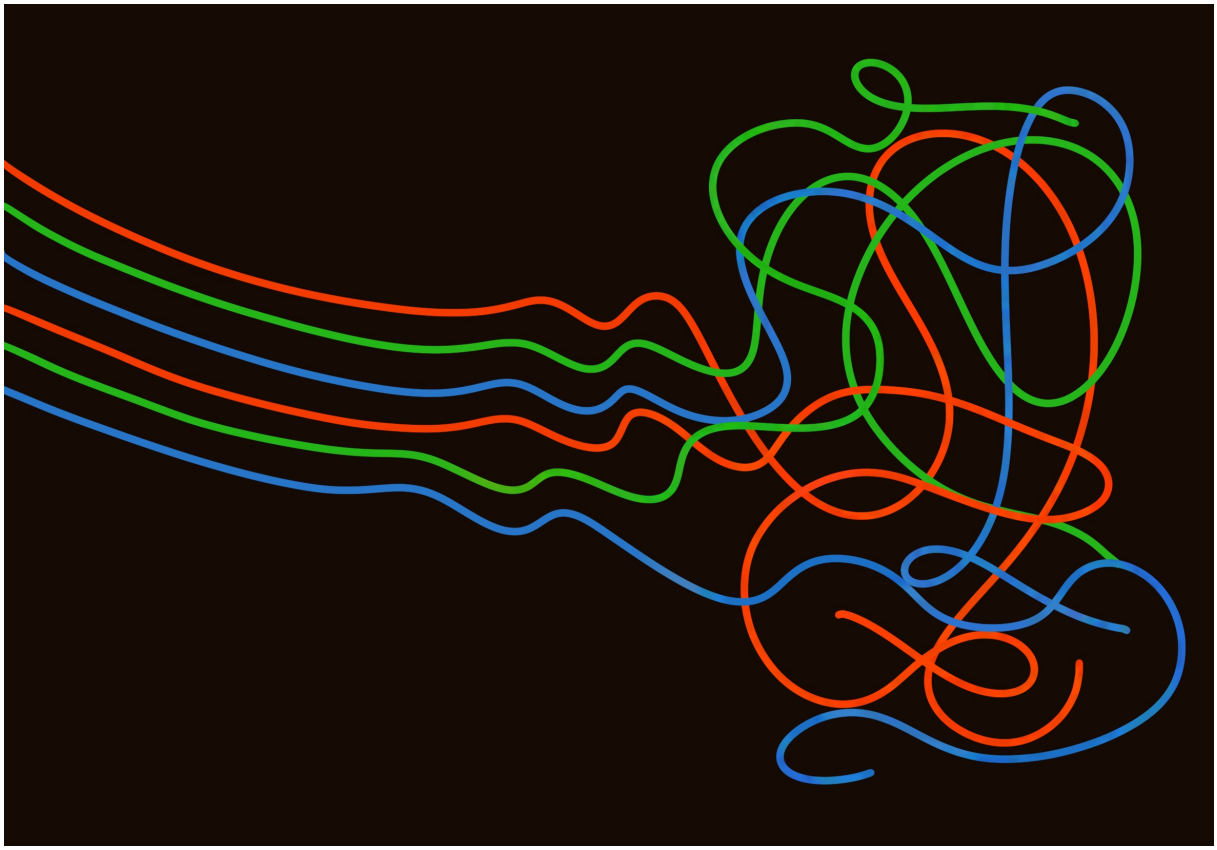

Figure 1: Artistic Rendition of Entropy

2 Entropy

The first fundamental concept we must understand is entropy itself. Entropy is a term used in many different way, originally developed in the context of thermodynamics and then formalised in the field of information theory by Claude Shannon in 1948 [1]. It provides a quantitative measure of uncertainty in a system, indicating how unpredictable the outcome of a random variable is. Formally, for a random variable $X \sim p(x)$, entropy quantifies the expected information content associated with the possible outcomes of X. Intuitively we can think of a less predictable variable having greater entropy. For example, consider a biased coin that lands head with probability 0.99. Since flipping a heads is overwhelmingly likely, we could predict a heads with high confidence - the uncertainty and entropy is low. On the other hand, a fair coin has no such uncertainty: both outcomes are equally likely, and the entropy is higher. Shannon defines entropy as the expected value of the information content, mathematically as follows.

Definition 2.1 (Shannon Entropy)**.** *Denoted as $H(X)$, the entropy of a discrete random variable $X \sim p(x)$ is defined as*

$$H(X) = -\sum_{x} p(x) \log_2(p(x)) \tag{1}$$

Note that, unless stated otherwise, log will denote base 2, and ln will denote the natural logarithm, base e. Furthermore, we use the convention that $0 \log 0 = 0$. In this expression, each term $p(x)$ represents the probability of observing each outcome x. The $\log p(x)$ terms weight the outcomes based on how likely they are: if $p(x)$ is close to 1, $\log p(x)$ will be close to zero. On the other hand, as $p(x)$ tends to 0, $\log p(x)$ tends to $-\infty$ - rarer variable contribute more to the overall entropy. As for all $p(x) \in [0, 1]$, $\log p(x)$ is negative, we multiply the expression by negative one to ensure entropy is a non-negative value. The derivation and further reasoning behind this formula will be explored as we continue.

Example 2.2. *What is the entropy of the coin that flips heads 99% of the time vs a fair coin?*

$$\begin{aligned}
H(X_{99}) &= -\sum_{k=0}^{1} p(k) \log(p(k)) \\
&= -\left(\frac{99}{100} \log \frac{99}{100} + \frac{1}{100} \log \frac{1}{100}\right) \\
&= 0.0808 \ \textit{(3 s.f.)} \\
H(X_{50}) &= -\sum_{k=0}^{1} p(k) \log(p(k)) \\
&= -\left(\frac{1}{2} \log \frac{1}{2} + \frac{1}{2} \log \frac{1}{2}\right) \\
&= 1
\end{aligned}$$

Entropy can also be thought of as the average amount of information required to describe the outcome of a random variable. This information is measured in bits, reflecting the number of binary decisions (or questions) are needed, on average, to specify an outcome. For example, a fair coin has two possible outcomes - say 0 for heads and 1 for tails - and therefore it has an entropy of 1. One binary question must be asked (e.g. "did you toss a tail?") to determine the outcome, and one bit is needed to describe the outcomes. We calculated this directly using the formula in Example (2.2).

More generally, for a uniform distribution over n outcomes (each with probability $\frac{1}{n}$), the entropy is given by $H(X) = \log n$ bits. This result is reasonably intuitive: $H(X)$ represents the number of bits required to identify an outcome, and $\lceil \log n \rceil$ is the maximum number of bits needed to encode n in binary. It is with non-uniform distributions where we face the less intuitive side of entropy.

Example 2.3 (Biased Die)**.** *What is the entropy of a die with probability distribution*

$$X \sim p(x), \quad p(x = (1,2,3,4,5,6)) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}\right)$$

The entropy of X follows as

$$\begin{aligned} H(X) &= -\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{8}\log\frac{1}{8} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{32}\log\frac{1}{32} + \frac{1}{32}\log\frac{1}{32}\right) \\ &= \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{8}\log 8 + \frac{1}{16}\log 16 + \frac{1}{32}\log 32 + \frac{1}{32}\log 32 \\ &= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{1}{4} + \frac{5}{32} + \frac{5}{32} = 1.9375 \text{ bits} \end{aligned}$$

This is telling us we need, on average, a minimum of 1.9375 bits to describe an outcome.

If we wanted to label each outcome sequentially, we would need 3 bits: 000, 001, 010, 011, 100, 101. If we were to cut any of the leading zeros, we would face ambiguity in messages, so all 3 bits are needed to guarantee correct data transmission. Clearly this is more than the entropy of the random variable, implying there may be a more optimal way to encode this data. Moreover, rolling a 1 is far more likely than rolling a 6; we may want to optimise our choices of bit length so that our ***average bit-length*** *is shorter. Consider the sequence 0, 10, 110, 1110, 11110, 11111. The expected (average) bit length of this sequence, given the probabilities defined above, is*

$$\frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{16}\cdot 4 + \frac{1}{32}\cdot 5 + \frac{1}{32}\cdot 5 = 1.9375$$

While we will mainly be considering the entropy of a single random variable, it is useful to outline a few related concepts that generalise the fundamentals of entropy to multiple variables. These ideas closely mirror probability functions, describing how uncertainty behaves in systems involving more than one random variable.

We may want to understand the combined or dependent uncertainty. The joint entropy measures the uncertainty for a joint distribution, like $(X, Y) \sim p(x, y)$. On the other hand, conditional entropy measures the entropy remaining in one random variable given that the value of another is already known.

Definition 2.4 (Joint Entropy)**.** *[3] Denoted as $H(X, Y)$, the joint entropy of a pair of discrete random variable $(X, Y) \sim p(x, y)$ is defined as*

$$H(X,Y) = -\sum_x \sum_y p(x,y)\log(p(x,y)) \tag{2}$$

Definition 2.5 (Conditional Entropy)**.** *[3] Denoted as $H(Y|X)$, the conditional entropy of a pair of discrete random variable $(X, Y) \sim p(x, y)$ is defined as*

$$H(Y|X) = -\mathbb{E}(\log p(Y|X)) \tag{3}$$

Where $\mathbb{E}$ denotes the expectation.

Theorem 2.6. *For a pair of discrete random variables $(X, Y) \sim p(x, y)$, we have*

$$H(X, Y) = H(Y|X) + H(X)$$

PROOF: From the conditional probability formula, we know

$$p(y|x) = \frac{p(x, y)}{p(x)} \Rightarrow p(x, y) = p(y|x)p(x)$$

Now, as defined above,

$$\begin{aligned} H(X) &= -\sum_x \sum_y p(x, y) \log(p(x, y)) \\ &= -\sum_x \sum_y p(x, y) \log p(y|x)p(x) \end{aligned}$$

Noting that $\log p(y|x)p(x) = \log p(y|x) + \log p(x)$, we can split the sums up:

$$\begin{aligned} &= -\sum_x \sum_y p(x, y) \log p(y|x)p(x) \\ &= -\sum_x \sum_y p(x, y) \log p(y|x) - \sum_x \sum_y p(x, y) \log p(x) \\ &= -\sum_x \sum_y p(x, y) \log p(y|x) - \sum_x p(x) \log p(x) \\ &= H(Y|X) + H(X) \end{aligned}$$

□

Beyond just measuring uncertainty, we may also wish to compare two distributions, or quantify how much information one variable provides another. The relative entropy measures how differently one probability distribution is from another, or how 'far apart' they are. It does not represent a true distance, as the triangle inequality is not satisfied, but it provides a useful indication of how much information is lost when one distribution is used to approximate another.

Definition 2.7 (Relative Entropy)**.** *[3] The relative entropy between two probability mass functions $p(x)$ and $q(x)$, denoted as $D(p||q)$ is given by*

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Where we take $0 \log \frac{0}{0} = 0$, $0 \log \frac{0}{q} = 0$, and $p \log \frac{p}{0} = \infty$.

We can also define the mutual information between two random variables. It measures the amount of information they share, or equivalently, how much knowig one variable reduces the uncertainty about the other.

Definition 2.8 (Mutual Information)**.** *[3] Given two random variables $(X, Y) \sim p(x, y)$, with corresponding marginal probability mass functions $p(x)$ and $p(y)$, the mutual information of the two random variables, denoted as $I(X; Y)$ is given as*

$$\begin{aligned} I(X; Y) &= \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= D(p(x, y)||p(x)p(y)) \end{aligned}$$

Theorem 2.9. *We have that*

- $I(X;Y) = I(Y;X)$
- $I(X;X) = H(X)$

Proof.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= I(Y;X) \end{aligned}$$

$$I(X;X) = H(X) - H(X|X) = H(X)$$

□

2.1 Derivation of the Shannon Entropy Formula

The formula for entropy defined in Definition 2.1 was first posed by Claude Shannon in his 1948 paper 'A Mathematical Theory of Communication' [1]. In the interest of completeness, we will derive the formula here as outlined in Appendix 2 of the above paper.

Derivation of $H(X) = -\sum_i p_i \log p_i$.

Our aim is to derive some way to quantify how much information a random variable holds, or how certain we are of an event. Consider a set of possible events $X = p_i,\ i = 1, \ldots, n$. We would like our formula $H(X)$ to satisfy the following properties:

1. $H(X)$ is continuous
2. Let $X = p_i$ for $i = 1, \ldots, n$ be a *uniform discrete random variable*, such that all $p_i = \frac{1}{n}$. Then $H(X)$ should be a *monotonously increasing function.* In other words, the more equally likely outcomes there are, the less certain we can be about which one will occur.
3. If an outcome can be split into 2 successive outcomes (e.g. rolling a 6 then a 2), H should be a weighted sum of the individual H's.

The third property of this list is the most important for our derivation. As such, we observe an example to help in our understanding of what this means. Consider the following two probability trees:

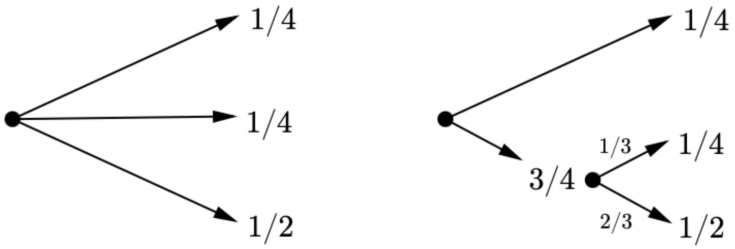

Figure 2: Two probability trees.

The left-hand tree in Figure 2 has 3 possible outcomes for one event, with probability $\frac{1}{4}, \frac{1}{4}$ and $\frac{1}{2}$ respectively. The right-hand has 2 events, $\frac{1}{4}$ chance of stopping immediately and $\frac{3}{4}$ of another outcome, leading to two more routes. These routes have probabilities $\frac{1}{3}$ and $\frac{2}{3}$ respectively. The key fact is that both trees end with the same probability distribution. As such, we will require that

$$H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right) = H\left(\frac{1}{4}, \frac{3}{4}\right) + \frac{3}{4}H\left(\frac{1}{3}, \frac{2}{3}\right)$$

Let's now consider a uniform discrete random variable X, and $H(X) = A(n)$ such that each outcome of X has a probability of $\frac{1}{n}$. We let $n = s^p$. This can be decomposed into p independent events of probability $\frac{1}{s}$, and as outlined in our third condition we get

$$A(s^p) = pA(s)$$

Similarly, setting $n = t^q$ we get

$$A(t^q) = qA(t)$$

We can choose p, q large enough to satisfy

$$s^p \leq t^q < s^{p+1}$$

which yields us

$$\begin{aligned} p \log s &\leq q \log t < (p+1) \log s \\ p &\leq q\left(\frac{\log t}{\log s}\right) < p + 1 \\ \frac{p}{q} &\leq \frac{\log t}{\log s} < \frac{p}{q} + \frac{1}{q} \end{aligned}$$

Given that q is arbitrarily large, we know that $\frac{1}{q} \to 0$ and we can rewrite this as

$$\left|\frac{p}{q} - \frac{\log t}{\log s}\right| < \varepsilon$$

for any $\varepsilon > 0$. From property 2, we know $A(n)$ is monotone, meaning

$$s^p \leq t^q < s^{p+1} \Rightarrow A(s^p) \leq A(t^q) < A(s^{p+1}) \Rightarrow pA(s) \leq qA(t) < (p+1)A(s)$$

Once again, through some rearrangement, we get

$$\begin{aligned}
pA(s) \leq qA(t) &< (p+1)A(s) \\
p \leq q\left(\frac{A(t)}{A(s)}\right) &< (p+1) \\
\frac{p}{q} \leq \frac{A(t)}{A(s)} &< \frac{p}{q} + \frac{1}{q} \\
\left|\frac{p}{q} - \frac{A(t)}{A(s)}\right| &< \varepsilon
\end{aligned}$$

Combining our two inequalities with ε, we get

$$\left|\frac{A(t)}{A(s)} - \frac{\log t}{\log s}\right| < 2\varepsilon \Rightarrow A(t) = k \log t, k > 0$$

Now we consider a set of n outcomes with probabilities $p_i = \frac{n_i}{\sum n_i}$, where we can consider each integer n_i to be some proportion of likelihood and $\sum n_i$ normalises it into a probability distribution we can manipulate. We decompose a set of $\sum n_i$ outcomes into a choice from n possibilities, with probabilities $p_1, \ldots, p_n$, and then a uniform choice from n_i possibilities.

For example, consider a bag with seven beads inside it. Three of the beads are red, two are blue, and two are green. We choose a bead from the bag at random, and we know that there is a $3/7$ chance we pull a red bead, $2/7$ chance we pull a blue bead and a $2/7$ chance we pull a green bead. But what is the probability of choosing a specific red bead? Clearly, we know it is $1/7$, but we can consider it another way: we have seven beads to choose from ($\sum n_i$), and we can first make a choice of the colour of the bead (n choices), and then choose a bead of that colour (n_i choices). In our example, we choose the colour red with probability $3/7$, and a specific red bead with probability $1/3$. $3/7 \times 1/3 = 1/7$ as required.

With this fact, we can use the third property once again to break up our entropy calculation:

$$\begin{aligned}
A\left(\sum n_i\right) &= \log \sum n_i \\
\log \sum n_i &= H(p_1, \ldots, p_n) + p_1(\log n_1) + p_2(\log n_2) + \cdots + p_i(\log n_i) \quad \textit{(see footnote)} \\
\log \sum n_i &= H(p_1, \ldots, p_n) + \sum p_i \log n_i \\
&\Downarrow \\
H(p_1, \ldots, p_n) &= \log \sum n_i - \sum p_i \log n_i = \log \sum n_i - \sum p_i \log n_i
\end{aligned}$$

Now we need to exploit a useful identity:

$$\begin{aligned}
p_i = \frac{n_i}{\sum n_i} \Rightarrow \log p_i &= \log\left(\frac{n_i}{\sum n_i}\right) = \log n_i - \log \sum n_i \\
-p_i \log p_i &= p_i(\log \sum n_i - \log n_i) && \text{(Multiplying through by } -p_i\text{)} \\
-\sum p_i \log p_i &= \sum p_i(\log \sum n_i - \log n_i) && \text{(Summing over } i\text{)} \\
-\sum p_i \log p_i &= \sum p_i \log \sum n_i - \sum p_i \log n_i \\
-\sum p_i \log p_i &= \log \sum n_i - \sum p_i \log n_i && \text{(Noting that } \textstyle\sum p_i = 1\text{)}
\end{aligned}$$

Note here that because this entropy calculation is not in terms of n, but in terms of p, where $X \sim p(x)$, we will use H instead of A.

Hence

$$H(p_1, \ldots, p_n) = \log \sum n_i - \sum p_i \log n_i = -\sum p_i \log p_i$$

as required. □

3 Optimisation Problems

The term *maximum entropy* inherently suggests an underlying optimisation problem. In essence, we are aiming to identify, among all possible probability distributions that satisfy a given set of constraints, the one that grants us the greatest entropy. We are transforming the concept of 'maximising uncertainty' into a well-defined problem.

Before we apply this to the entropy formula, we will first review the general structure of optimisation problems. We begin by introducing the fundamentals of optimisation, and build the foundations we will need to apply these ideas to maximum entropy models.

Definition 3.1. *An optimisation problem takes the the following form:*

$$\begin{aligned} &\textbf{\textit{Minimise}}\ f(\boldsymbol{x}) && \text{for } \boldsymbol{x} \in \mathbb{R}^n \\ &\textbf{\textit{Subject to}}\ g_i(\boldsymbol{x}) = 0 && \text{for } i = 1, \dots, m \\ &\qquad\qquad h_j(\boldsymbol{x}) \leq 0 && \text{for } j = 1, \dots, k \end{aligned} \tag{4}$$

Here, $f(\boldsymbol{x})$ is the objective function, it is the object we want to minimise. The $g_i(\boldsymbol{x}) = 0$ are the equality constraints, the $h_j(\boldsymbol{x}) \leq 0$ are the inequality constraints, and $\boldsymbol{x}$ is the optimisation variable. We will generally denote this without the boldface, but it is important to remember that x is a vector.

The domain D of an optimisation problem is the intersection of all individual domains in the problem. The set of points in D that satisfy all the equality and inequality constraints of the problem is called the **feasible** set. The problem is said to be **feasible** if this set is non-empty. The **optimal value**, p^* is defined as

$$p^* = \inf\{f(x) \mid g_i(x) = 0 \text{ for } i = 1, \dots, m,\ h_j(x) \leq 0 \text{ for } j = 1, \dots, k\}$$

A point x^* that satisfies all constraints and attains the optimal value, i.e. $f(x^*) = p^*$, is referred to as an **optimal point**.The set of all optimal points, X_{opt}, is given by

$$X_{\text{opt}} = \{x^* \mid f(x^*) = p^*,\ g_i(x^*) = 0 \text{ for } i = 1, \dots, m,\ h_j(x^*) \leq 0 \text{ for } j = 1, \dots, k\}.$$

By convention, any optimisation problem of the form (4) is treated as a minimisation problem. In order to maximise a function $f(x)$, we simply minimise its negation, $-f(x)$.

3.1 Convexity

One of the most useful and pertinent forms of optimisation problems is convex optimisation, where our objective function and constraints satisfy certain convexity conditions. Convex problems are particularly important because their solutions are guaranteed to exist and be unique, which greatly simplifies both analysis and the methods used to solve them. Before exploring these problems in detail, we take a brief aside to reaffirm the formal definition of convexity [11].

Definition 3.2 (Convex Set)**.** *A set C is convex if the line segment between any two points in C lies entirely in C. We can formalise this by saying for any two points $c_1, c_2 \in C$, and any $\theta : 0 \leq \theta \leq 1$ we have*

$$\theta c_1 + (1 - \theta)c_2 \in C$$

.

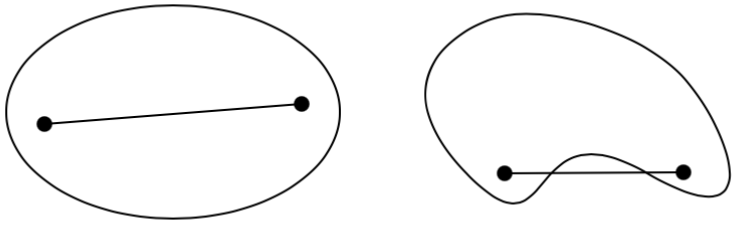

Figure 3: Left: convex set, Right: Non-convex set

It is important to note that, for a set to be convex, only the line segment connecting any two points must lie entirely in the set. Clearly, in Figure 3, the set on the left-hand side is bounded: if we were to extend the line infinitely, we would leave the boundary of the set. To generalise this concept to include the entire line, we introduce the notion of affine sets.

Definition 3.3 (Affine Set)**.** *A set C is affine if the line that connects any two points in C lies entirely in C. We can formalise this by saying for any two points $c_1, c_2 \in C$, and any $\theta \in \mathbb{R}$ we have*

$$\theta c_1 + (1 - \theta)c_2 \in C$$

All affine sets are convex, but not all convex set are affine.

We can also introduce the closely related concepts of affine and convex functions

Definition 3.4 (Affine Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if it has the form $f = Ax + b$, where A is a linear function such that $A \in \mathbb{R}^{m \times n}$, and b is a constant vector in $\mathbb{R}^m$.*

Let $S \subseteq \mathbb{R}^n$ be a convex set, and $f : S \to \mathbb{R}^m$ is an affine function. Then the image of S under f is convex:

$$f(S) = \{f(x) \mid x \in S\} \text{ is convex}$$

Definition 3.5 (Convex Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^n$, and for all $\theta : 0 \leq \theta \leq 1$ we have*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) \tag{5}$$

A function is strictly convex if the strict inequality holds in (5). A function is concave if $-f$ is convex, and strictly concave if $-f$ is strictly convex.

Geometrically, this means that the chord between any two points on the function f lies above f

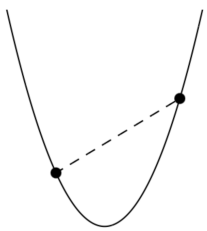

Figure 4: A convex function

There are several properties that can be used to determine if a function is convex. Consider the first order condition:

Theorem 3.6 (First Order Condition for Convexity)**.** *Suppose f is differentiable. Then f is convex if* $\operatorname{dom} f$ *is convex and*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

hold for all $x, y \in \operatorname{dom} f$. This is the first order Taylor expansion of f, and a function is convex if and only if this expansion is a global under-estimator.

This is a key property of convex functions, and one that we will exploit as we continue.

Proof. We first consider the case $n = 1$. We will show that a differentiable $f : \operatorname{dom} f \to \mathbb{R}$ is convex if and only if

$$f(y) \geq f(x) + f'(x)(y - x)$$

($\Rightarrow$)Let f be convex for all $x, y \in \operatorname{dom} f$. We know that $\operatorname{dom} f$ must be a convex set, as it is either $\mathbb{R}$ or an interval (as f is differentiable), and we can say that for all $0 \leq \theta \leq 1$ we have $\theta y + (1 - \theta)x = x + \theta(y - x) \in \operatorname{dom} f$. By the convexity of f, we know

$$\begin{aligned}
f(\theta y + (1-\theta)x) &\leq \theta f(y) + (1-\theta) f(x) \\
f(x + \theta(y - x)) &\leq f(x) + \theta(f(y) - f(x)) && \text{(Divide through by } \theta) \\
f(y) - f(x) &\geq \frac{f(x + \theta(y-x)) - f(x)}{\theta} && \text{(Take the limit as } \theta \to 0) \\
f(y) &\geq f(x) + f'(x)(y - x)
\end{aligned}$$

($\Leftarrow$) Assume $f(y) \geq f(x) + f'(x)(y - x)$. Choose any $x, y \in \operatorname{dom} f : x \neq y$. Let $z = \theta x + (1 - \theta)y$ for some $\theta : 0 \leq \theta \leq 1$. Applying our assumed condition twice, we can yield

$$f(x) \geq f(z) + f'(z)(z - x) \qquad\qquad f(y) \geq f(z) + f'(z)(z - y)$$

Multiplying the left hand inequality by θ and the right hand by $(1 - \theta)$, we can sum the two to yield

$$\begin{aligned}
\theta f(x) + (1-\theta) f(y) &\geq \theta f(z) + \theta f'(z)(z - x) + (1-\theta) f(z) + (1-\theta) f'(z)(z - y) \\
f(z) &\leq \theta f(x) + (1-\theta) f(y) \\
f(\theta x + (1-\theta)y) &\leq \theta f(x) + (1-\theta) f(y)
\end{aligned}$$

as required. For our higher order cases, let $f : \mathbb{R}^n \to \mathbb{R}$, and $x, y \in \mathbb{R}^n$. Consider $g(\theta) = f(\theta y + (1-\theta)x)$ for $0 \leq \theta \leq 1$. By the chain rule we can say that $g'(\theta) = \nabla f(\theta y + (1-\theta)x)^T (y - x)$. Essentially, we are mapping our n-dimension function back into 1-dimension, and using the previously proven condition.

($\Rightarrow$) Assume f is convex, which in turn implies g is convex. We have $g(1) \geq g(0) + g'(0)$ from the $n = 1$ case. This implies we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

as required.

($\Leftarrow$) Assume for all $x, y \in \operatorname{dom} f$, we have $f(y) \geq f(x) + \nabla f(x)^T (y - x)$. We also keep $g(\theta)$ as defined above. We want to show that g is convex, which would imply f is convex. Choose any $\theta, \varphi \in [0, 1]$, such that $\theta y + (1 - \theta)x \in \operatorname{dom} f$ and $\varphi y + (1 - \varphi)x \in \operatorname{dom} f$. We will have

$$f(\theta y + (1-\theta)x) \geq f(\varphi y + (1-\varphi)x) + \nabla f(\varphi y + (1-\varphi)x)^T (y - x)(\theta - \varphi)$$

or, equivalently,

$$g(\theta) \geq g(\varphi) + g'(\varphi)$$

implying g is convex, and therefore f in convex as required. □

This inequality is sometimes referred to as Jensen's inequality, and can be extended to more than two points. For a convex function f, $\theta_1, \ldots, \theta_k \geq 0$ and $\theta_1 + \cdots + \theta_k = 1$ we have

$$f(\theta_1 x_1 + \cdots + \theta_k x_k) \geq \theta_1 f(x_1) + \cdots + \theta_k f(x_k)$$

A convex optimisation problem has a form very similar to (4), but includes a few additional requirements and subtle distinctions.

Definition 3.7. *A convex optimisation problem takes the form*

$$\begin{aligned} &\textbf{\textit{Minimise}}\ f(x) && \text{for } x \in \mathbb{R}^n \\ &\textbf{\textit{Subject to}}\ a_i^T(x) = b_i && \text{for } i = 1, \ldots, m \\ &\qquad\qquad h_j(x) \leq 0 && \text{for } j = 1, \ldots, k \end{aligned} \tag{6}$$

Where we note the following requirements:

1. *The objective function $f(x)$ must be convex.*

2. *The inequality constraints $h_j(x)$ must all be convex.*

3. *The equality constraint functions $g_i(x) = a_i^T x - b_i$ must be affine.*

3.2 Lagrange Multipliers

In optimisation problems, the variables are subject to constraints that must be satisfied, Ignoring these constraints and choosing values arbitrarily is an inefficient way to determine the maximum or minimum. The method of Lagrange multipliers provides a systematic way to incorporate these constraints while finding extrema of a function. Essentially, we will convert a constrained optimisation problem into a form that can be manipulated with standard techniques for optimisation problems.

To illustrate this idea, consider the following optimisation problem.

$$\begin{aligned} &\textbf{Minimise}\ f(x) && \text{for } x \in \mathbb{R}^n \\ &\textbf{Subject to}\ g_i(x) = 0 && \text{for } i = 1, \ldots, m \end{aligned} \tag{7}$$

The goal is to find some $x \in \mathbb{R}^n$ that minimises $f(x)$ while satisfying each additional constraint, $g_i(x) = 0$. To do this, we introduce the Lagrangian, a function that combines the original objective function and the constraints using Lagrange multipliers.

Definition 3.8 (Lagrangian)**.** *[2] For an optimisation problem (7), we define*

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

For all $x \in \mathbb{R}^n$

Theorem 3.9. *The solution to the optimisation problem (7) will be $x \in \mathbb{R}^n, \lambda \in \mathbb{R}^m$ satisfying*

$$\frac{\partial L}{\partial x_j} = 0, \ \frac{\partial L}{\partial \lambda_i} = 0 \ \ \textit{for } j = 1, \ldots, n, \ i = 1, \ldots, m.$$

Example 3.10. *In this example, we want to maximise the area of a rectangle under the graph* $x + y = 1$.

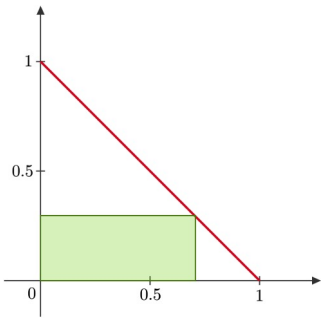

Figure 5: Rectangle under the graph $x + y = 1$, bounded by the axes.

This is a reasonably trivial problem, but it serves as a basis of understanding as we continue to more complicated questions. Let x *be the* $x-$*intercept and* y *be the* $y-$*intercept. The area of the rectangle is therefore* xy*. When we deal with the Lagrangian, we always aim to minimise, so instead of maximising the area, we will minimise the negative area, ie* $-xy$*. We have* $x = (x, y)$ *and* $g(x) = x + y - 1$*, as we need* $g(x) = 0$.

$$\begin{aligned} L(x, \lambda) &= -xy + \lambda(x + y - 1) \\ \frac{\partial L}{\partial x} &= -y + \lambda \\ \frac{\partial L}{\partial y} &= -x + \lambda \\ \frac{\partial L}{\partial \lambda} &= x + y - 1 \end{aligned}$$

Setting each of these derivatives equal to zero, we get

$$x = y = \lambda,\ 2\lambda - 1 = 0 \Rightarrow x = y = \lambda = \frac{1}{2}$$

So our rectangle of maximum area is the square $[0, \frac{1}{2}] \times [\frac{1}{2}, 0]$

3.3 Dual Function

As we noted earlier in this chapter, not all optimisation problems have the exact structure of (7). Many problems involve both inequality and equality constraints, and the definition of the Lagrangian can be extended to include them.

Definition 3.11 (Lagrangian 2)**.** *Consider the optimisation problem*

$$\begin{aligned} &\textbf{\textit{Minimise}}\ f(\boldsymbol{x}) && \text{for } \boldsymbol{x} \in \mathbb{R}^n \\ &\textbf{\textit{Subject to}}\ g_i(\boldsymbol{x}) = 0 && \text{for } i = 1, \ldots, m \\ &\qquad\qquad\ h_j(\boldsymbol{x}) \leq 0 && \text{for } j = 1, \ldots, k \end{aligned} \tag{8}$$

We define the Lagrangian of this optimisation problem as

$$L(x, \lambda, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \nu_i h_i(x)$$

We refer to λ_i *as the Lagrangian multiplier of the* i*-th equality constraint, and* ν_j *as the Lagrangian multiplier of the* j*-th inequality constraint. We note that* $\lambda_i \geq 0$ *to ensure valid lower bounds. The vectors* λ *and* ν *are referred to as the dual variables of the optimisation problem.*

Definition 3.12 (Lagrangian Dual Function)**.** *We define the Lagrangian dual function (or simply dual function) as the minimum value of the Lagrangian over x:*

$$G(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left(f(x) + \sum_i \lambda_i g_i(x) + \sum_j \nu_i h_i(x) \right)$$

Where D is the domain of the optimisation problem, $\lambda \in \mathbb{R}^m, \nu \in \mathbb{R}^k$.

The Lagrangian dual function provides a lower bound for p^*, the optimal value, whenever $\lambda \geq 0$. In other terms, for any pair $\lambda \geq 0, \nu$ we have

$$G(\lambda, \nu) \leq p^*$$

This naturally leads to the question of which pair (λ^*, ν^*) is the 'best'. That is, which values maximise the lower bound of G

Definition 3.13 (Lagrange Dual Problem)**.** *The Lagrange dual problem is the optimisation problem*

$$\begin{aligned} \textbf{\textit{Maximise}} \quad & G(\lambda, \nu) \\ \textbf{\textit{Subject to}} \quad & \lambda \geq 0 \end{aligned} \tag{9}$$

We refer to the pair (λ^*, ν^*) as the dual optimal or optimal Lagrangian multipliers if they are optimal for (9). The dual problem is a convex optimisation problem, regardless of whether or not the original optimisation problem is.

Definition 3.14 (Weak Duality)**.** *The optimal value d^* of the Lagrange dual problem is the best lower bound of p^*. We have that*

$$d^* \leq p^*$$

even if the original problem is not convex. This is weak duality. The term $p^* - d^*$ is known as the optimal duality gap, and is always non-negative

Definition 3.15 (Strong Duality)**.** *If we have that $d^* = p^*$, i.e., the optimal duality gap is zero, then we have strong duality.*

We do not have strong duality in general. If the original problem is convex, we *usually* have strong duality, but this is not guaranteed. It is therefore useful to establish conditions under which strong duality will hold, which are called constraint qualifications. One such commonly used is Slater's condition:

Theorem 3.16 (Slater's Condition)**.** *If a convex optimisation problem has a 'strictly' feasible point, that is, a feasible point that satisfies all inequality constraints strictly, then strong duality holds. In other terms, for the convex optimisation problem*

$$\begin{aligned} \textbf{\textit{Minimise}} \ & f(x) && \text{for } x \in \mathbb{R}^n \\ \textbf{\textit{Subject to}} \ & a_i^T(x) = b_i && \text{for } i = 1, \ldots, m \\ & h_j(x) \leq 0 && \text{for } j = 1, \ldots, k \end{aligned}$$

if there exists an $\hat{x} \in \operatorname{dom} f$ such that

- *For all $i = 1, \ldots, m$, $a_i^T(\hat{x}) = b_i$*
- *For all $j = 1, \ldots, k$, $h_j(\hat{x}) < 0$*

then strong duality holds, and $d^* = p^*$.

3.4 KKT Conditions

Our main goal now is to understand a systematic method to solve an optimisation problem with both equality and inequality constraints. The methodology we will consider is known as the Karush–Kuhn–Tucker (KKT) conditions, which generalise Lagrangian multipliers. They provide necessary - and for a convex problem, sufficient - conditions for an optimal point. Consider a general problem:

$$\begin{aligned} \textbf{Minimise } & f(x) && \text{for } x \in \mathbb{R}^n \\ \textbf{Subject to } & g_i(x) = 0 && \text{for } i = 1, \dots, m \\ & h_j(x) \leq 0 && \text{for } j = 1, \dots, k \end{aligned}$$

which has the Lagrangian

$$L(x, \lambda, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \nu_i h_i(x)$$

The KKT conditions are as follows.

Theorem 3.17 (KKT Conditions)**.** *For a point x^* to be optimal, there must exist multipliers λ_i and ν_j such that the following hold.*

1. ***(Stationary)*** *$\nabla_x L = 0$, or $\frac{\partial L}{\partial x_i} = 0$ for $i = 1, 2, \dots, n$.*
2. ***(Primal Feasibility)*** *$g_i(x^*) = 0$ for $i = 1, \dots, m$, $h_j(x) \leq 0$ for $j = 1, \dots, k$.*
3. ***(Dual Feasibility)*** *$\nu_j \geq 0$.*
4. ***(Complimentary Slackness)*** *$\nu_j h_j(x^*) = 0$. This implies that either $\nu_j = 0$, or $h_j(x^*) = 0$. If $h_j(x) = 0$, we call the inequality constraint 'active' and it is 'inactive' otherwise.*

4 Maximising Entropy

We now aim to explore how we can use Lagrange multipliers to maximise the entropy formula (Definition 2.1). Let us first consider a basic case.

4.1 Base Case

$$\textbf{Maximise } H(X) = -\sum_{i}^{n} p_i \log(p_i) \quad \text{for } X = (p_1, \ldots, p_n) \in \mathbb{R}^n$$

$$\textbf{Subject to } \sum_{i}^{n} p_i = 1$$

$$p_i \geq 0$$

For this example, we are looking at a very basic maximum entropy problem, where we have only constrained the optimisation with the fundamental laws of probability (all probabilities must be positive, and sum to one). We will assemble a Lagrangian of this problem, and then use KKT conditions to solve it.

As we explored in Chapter 3, we will want to transform our maximisation problem into a minimisation one. Rewriting this, we are left with

$$\textbf{Minimise } H(X) = \sum_{i}^{n} p_i \log(p_i) \quad \text{for } X = (p_1, \ldots, p_n) \in \mathbb{R}^n$$

$$\textbf{Subject to } \sum_{i}^{n} p_i = 1$$

$$p_i \geq 0$$

We can also check that our entropy formula is convex, which will aid in our conclusions. Consider

$$g(x) = x \log x = \frac{1}{\ln 2} x \ln x$$

. We have swapped into base e for simplicity when we differentiate. We have

$$g'(x) = \frac{1}{\ln 2}\left(x \cdot \frac{1}{x} + 1 \cdot \ln x\right) = \frac{1}{\ln 2}(1 + \ln x)$$

$$g''(x) = \frac{1}{\ln 2} \cdot \frac{1}{x} > 0 \quad \text{for } x \in (0, \infty)$$

By our convention that $0 \log 0 = 0$, $g(0)$ is well defined and we can conclude that $g(x)$ is convex on $[0, \infty)$. As $H(X)$ is the sum of n convex functions, it must also be convex. We can also conclude that our constraints must be convex.

Now we must formulate our Lagrangian. We have

$$L(p, \lambda, \mu) = \sum_{i}^{n} p_i \log p_i + \lambda\left(\sum_{i}^{n} p_i - 1\right) - \sum_{i}^{n} \mu_i p_i$$

Note that we must take away the $\mu_i p_i$ term, as in our problem (4.1), we have $p_i \geq 0$, whereas in standard optimisation problems, the inequality constraints are below zero.

Our first step of the KKT conditions is to find the partial derivatives of L with respect to each p_i. We have

$$L(p_i, \lambda, \mu_i) = p_i \log p_i + \lambda(p_i - 1) - \mu_i p_i$$
$$\frac{\partial L}{\partial p_i} = \frac{1}{\ln 2}(1 + \ln p_i) + \lambda - \mu_i$$

Setting this equal to zero, we get

$$\begin{aligned}
\frac{\partial L}{\partial p_i} &= \frac{1}{\ln 2}(1 + \ln p_i) + \lambda - \mu_i = 0 \\
\ln p_i &= \ln 2(\lambda - \mu_i) - 1 \\
p_i &= \exp(\ln 2(\lambda - \mu_i) - 1) \qquad (*)
\end{aligned}$$

Which we will simplify further in due course. We now must consider primal feasibility and dual feasibility: our minimum must satisfy the conditions we have outlined, i.e.

$$\sum_i^1 p_i = 1, \quad p_i \geq 0$$

and we must have that $\mu_i \geq 0$. We also have complimentary slackness, which tells us that $\mu_i p_i = 0$. As such, if $p_i \neq 0$, we must have $\mu_i = 0$.

For now, let us assume that all $p_i > 0$, and all other conditions hold. This means that all $\mu_i = 0$, so $(*)$ simplifies to

$$\begin{aligned}
p_i &= \exp(\lambda \ln 2 - 1) = \exp(\ln 2^\lambda - 1) \\
&= \exp(\ln 2^\lambda) \cdot \exp(-1) \\
&= \frac{2^\lambda}{e}
\end{aligned}$$

Which does not depend on i. This tells us that at our optimised point, all p_i must be equal. As such, let $p_i = c$. We can apply this to our other conditions, namely

$$\begin{aligned}
\sum_i^n p_i &= 1 \\
\sum_i^n c &= 1 \\
nc = 1 &\Rightarrow c = \frac{1}{n}
\end{aligned}$$

This means that, in this simple case where we have only defined the normal rules of probability, our maximum entropy $H(X)$ occurs when

$$X = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$$

Intuitively, this makes sense. When we maximise entropy, we aim to find the distribution that makes the fewest assumptions. A uniform distribution like X does not favour any particular outcome, and assumes that all are as equally likely.

4.2 Expected Value Equality Constraint

We now aim to find our maximum entropy with an additional equality constraint. Expectation is a fundamental in probability and statistics, so it is logical to begin there. Our optimisation problem is as follows.

$$\begin{aligned}
\textbf{Minimise } H(X) &= \sum_i^n p_i \log(p_i) \quad \text{for } X = (p_1, \ldots, p_n) \in \mathbb{R}^n \\
\textbf{Subject to } &\sum_i^n p_i = 1 \\
&\sum_i^n p_i x_i = \mu \\
&p_i \geq 0
\end{aligned}$$

As we did previously, it will aide our process if we rewrite our objective function in terms of natural logs, yielding

$$\begin{aligned}
\textbf{Minimise } H(X) &= \frac{1}{\ln 2} \sum_i^n p_i \ln(p_i) \quad \text{for } X = (p_1, \ldots, p_n) \in \mathbb{R}^n \\
\textbf{Subject to } &\sum_i^n p_i = 1 \\
&\sum_i^n p_i x_i = \mu \\
&p_i \geq 0
\end{aligned}$$

Our next step is to form our Lagrangian. We have

$$\begin{aligned}
L(p, \lambda, \beta, \alpha) &= \frac{1}{\ln 2} \sum_i^n p_i \ln(p_i) + \lambda \left(\sum_i^n p_i - 1 \right) + \beta \left(\sum_i^n p_i x_i - \mu \right) - \sum_i^n \alpha_i p_i \\
&= \frac{1}{\ln 2} \sum_i^n p_i \ln(p_i) + \sum_i^n p_i (\lambda + \beta x_i - \alpha_i) - \lambda - \beta\mu
\end{aligned}$$

Note here that μ signifies the expected value as is standard, and $\lambda,\ \alpha$ and β are the Lagrangian multipliers. Taking our derivatives, we get

$$\begin{aligned}
\frac{\partial L}{\partial p_i} &= \frac{1}{\ln 2}(\ln p_i + 1) + \lambda + \beta x_i - \alpha_i = 0 \\
\ln p_i &= \ln 2(\alpha_i - \beta x_i - \lambda) - 1 \\
p_i &= 2^{\alpha_i - \beta x_i - \lambda} e^{-1} \\
&= \frac{1}{e}(2^{\alpha_i - \beta x_i - \lambda})
\end{aligned}$$

We note our other conditions: $\sum p_i = 1,\ \sum p_i x_i = \mu,\ p_i \geq 0$ and complimentary slackness, $\alpha_i p_i = 0$. Consider the case where p_i is in active support (ie $p_i \neq 0$). This forces all $\alpha_i = 0$, and

our stationary point simplify to

$$p_i = c(2^{-\beta x_i})$$

where $c = \frac{2^{-\lambda}}{e}$. If $p_i = 0$, we would have the term $(\ln 0 + 1) = -\infty$, so we discard the boundary and set the condition that $p_i > 0$. To find c, let us consider the normalisation term $\sum p_i = 1$.

$$\sum_i^n p_i = \sum_i^n c(2^{-\beta x_i}) = 1$$
$$c = \frac{1}{\sum 2^{-\beta x_i}}$$

Finding β is harder in practice, and will usually need to be done numerically. Let $\sum 2^{-\beta x_i} = g(\beta)$ We have

$$\sum_i^n \frac{2^{-\beta x_i}}{g(\beta)} x_i = \mu$$
$$\sum_i^n 2^{-\beta x_i} x_i = \mu g(\beta)$$

where β is the solution to the above equation for given μ and x_i. Our final result, and our optimised set of probabilities comes out as

$$p_i = \frac{2^{-\beta x_i}}{g(\beta)} \quad \text{where} \quad p_i > 0, \ \sum_i^n 2^{-\beta x_i} x_i = \mu g(\beta)$$

5 Example Model and Analysis: Ariolimax

Figure 6: Ariolimax, more commonly known as the Banana Slug. Image credits: Ben Stanfield [10]

5.1 Understanding Elapid

One of the most interesting applications of maximum entropy lies in species distribution models (SDMs). The key idea behind applying a maximum entropy model (or MaxEnt, in modelling literature) to an SDM is to evaluate what environmental factors influence the occurrence of a species in a particular location, and use this information to predict where else the species might occur. In essence, if we know where a species has been observed, and we have environmental data about that location, we could estimate where similar conditions exist, and hence where the species may be found.

We will feed the model two types of data: a set of environmental variables, and some occurrence records. The MaxEnt model then tries to predict a probability distribution that estimates the environmental suitability of a location for the species, subject to the condition that we maximise entropy, and don't over-fit the data. Importantly, as we only have presence data (and not 'absence' data), the model will indicate the relative likelihood of an area of land hosting the species, rather than a strict probability of presence.

We will implement this approach in Python, primarily using the Elapid package [5]. Elapid is designed primarily for SDMs, and implements a tailored geospatial MaxEnt modelling system. It aims to estimate the probability distribution of the environmental suitability for the species in the least biased way possible - by maximising entropy. We outline a basic understanding of the MaxEnt modelling process in Elapid.

1. Input Data.

   The first step is to give the model the correct data to work with. We must input an occurrence dataset: known locations where the species has been observed (generally expressed as decimal latitude-longitude coordinates).

   We also must give a set of environmental features, such as temperature, precipitation or

topography, that may influence species distribution. These are represented as rasters, geospatial datasets in image form where each pixel encodes the data for certain area, or land parcel. The resolution of the rasters (i.e. the number of pixels in the image) will determine the *size* of each land parcel.

The model associates each occurrence point with its environmental features, and stores them in a 'feature vector'

$$x_i = (\text{temperature}_i,\ \text{precipitation}_i, ...)$$

To represent the available environment, Elapid also samples additional background points, known as pseudo-absences, since it has true absence data.

2. Transforming the features.

   Next, Elapid applies feature transformations to capture the more complex relationships between species presence and environmental variables [5][4]. These transformations include:

   - Linear features, a simple proportional relationship
   - Quadratic features, a hump shaped response that distinguishes between central values and extremes
   - Product features, the interaction between several variables
   - Hinge features, piecewise thresholds

   These transformations improve the predictive accuracy of the model by allowing non-linear and interaction effects between predictors.

3. Maximum Entropy Optimisation Problem

   The model will then solve the MaxEnt optimisation problem:

$$\textbf{Maximise } H(X) = -\sum_{i}^{n} p_i \log(p_i) \quad \text{for } X = (p_1, \ldots, p_n) \in \mathbb{R}^n$$

$$\begin{aligned} \textbf{Subject to } & \sum_{i}^{n} p_i = 1 \\ & \sum_{x} p(x) f_j(x) = \bar{f}_j \qquad \text{for all } j \\ & p_i \geq 0 \end{aligned}$$

   Here, $f_j(x)$ are the features discussed, and $\bar{f}_j$ is the average of that feature across the presence points.

   Solving using the Lagrangian methods yield the probability distribution

$$p(x) = \frac{1}{z} e^{(\sum \beta_j f_j(x))}$$

   where z is the normalisation constant.

4. Regularisation

   To prevent over-fitting, Elapid adds a regularisation term. This reduces weights overall, unless there is strong support for important feature. This is especially important for small sample sets, correlated environmental factors and any bias that may occur in the data.

5. Prediction

   Finally, the fitted model produces suitability predictions across the study region, normalised to a probability distribution. It will then create a raster, with each pixel coloured for the predicted suitability for the species to occur. This can be mapped for an interpretable visualisation of the model's results.

5.2 Mapping Ariolimax Occurrences

To demonstrate how maximum entropy models perform in practice, we consider a real-world example. We aim to analyse the distribution of various species of Ariolimax, more commonly known as banana slugs, found on the Pacific coast of North America. These slugs typically inhabit the temperate, moist forest floors of the west coast [12].

We will train a maximum entropy model on a large dataset on known Ariolimax occurrence locations (in decimal latitude and longitude) recorded between 2005 and 2023 [7]. Throughout this section, we will refer to this as the point data. Additionally, we employ 19 environmental rasters: the standard WorldClim Bioclimatic variables [13] at a resolution of 2.5 arc-minutes. This equates to a grid of land parcels where each cell covers approximately 4.6km^2.

We first visualise the known occurrences of each Ariolimax species. This helps is identify broad spatial patterns and provides a baseline for later comparison with our model predictions.

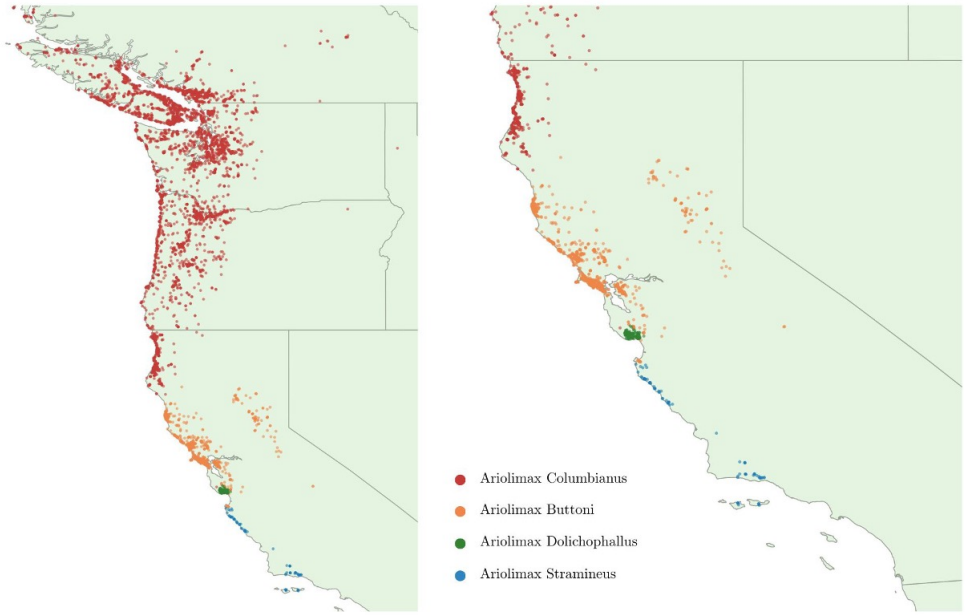

Figure 7: Left: Map of each species as given in the legend in the research area. Right: Map of each species as given in the legend, zoomed into California.

Distinct spatial trends emerge clearly in this plot. In the northernmost regions, the Ariolimax Columbianus has an obvious monopoly. They extending south into the north of California, where the other species begin to appear. Several regions of California show overlapping species

populations, which may be potential areas of competition that will warrant closer examination later.

We next examine the environmental variables used to train the model. In this example, we primarily consider temperature and precipitation data averaged over 1970-2000. These climatic features are widely considered to be key in species distribution. As shown below, the Ariolimax species exhibit a clear tendency towards regions with moderate temeratures and high rainfall.

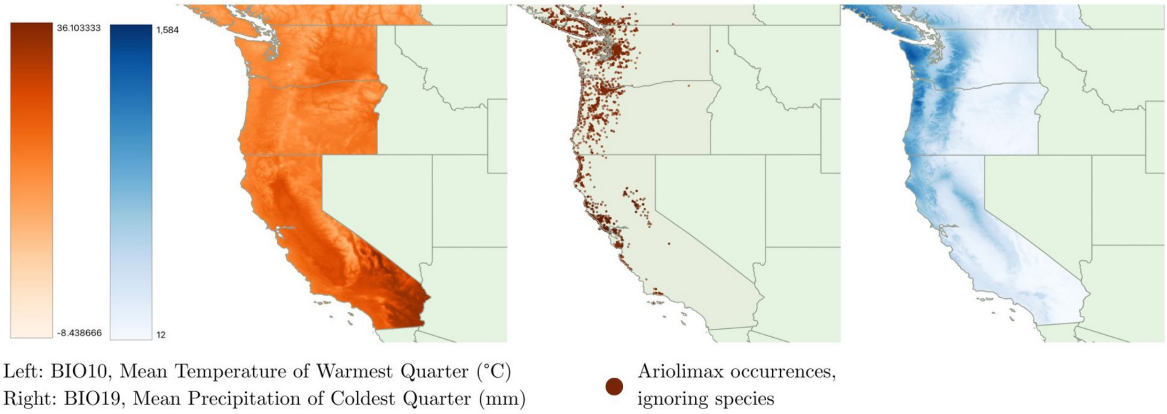

Figure 8: Left: Map of BIO10, Mean Temperature of Warmest Quarter (deg C). Middle: Map of Ariolimax, ignoring species. Right: Map of BIO19, Mean Precipitation of Coldest Quarter (mm).

Ariolimax Columbianus MaxEnt Model

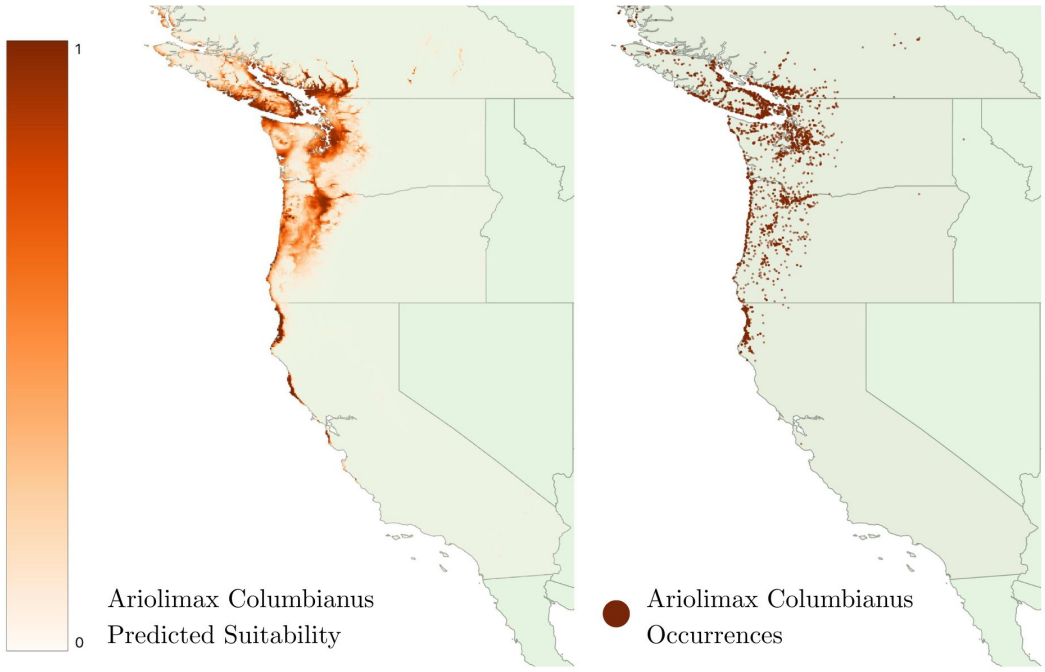

Figure 9: Left: Map of the MaxEnt model predicted suitability for the Ariolimax Columbianus over the research area. Right: The known occurrences of the Ariolimax Columbianus over the research area.

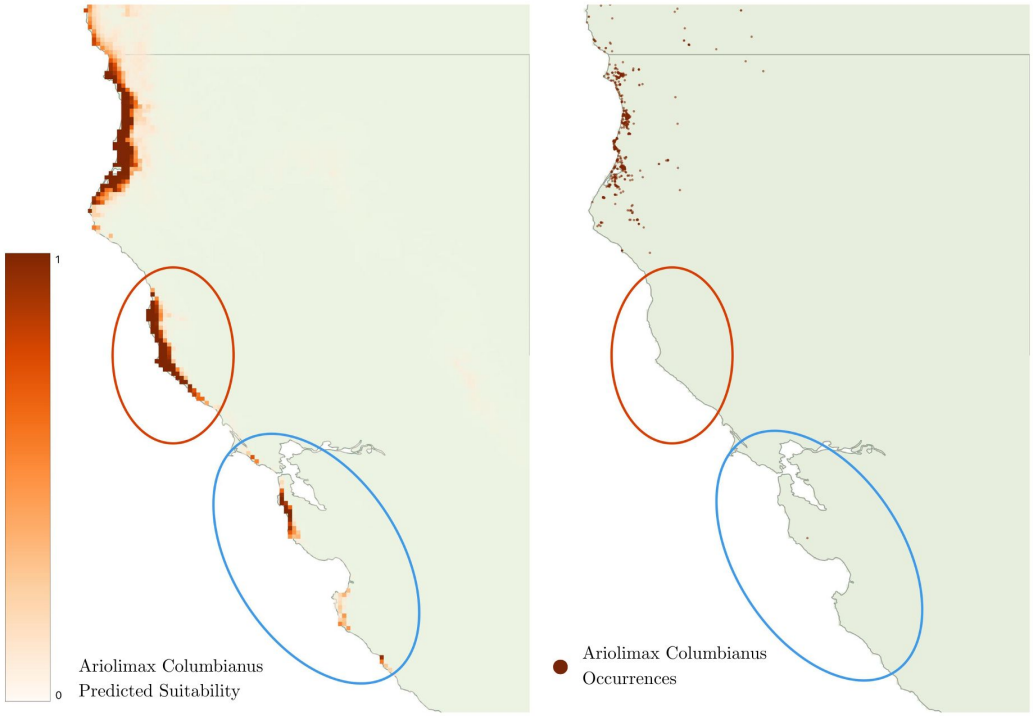

Figure 10: Left: Map of the MaxEnt model predicted suitability for the Ariolimax Columbianus zoomed into California. Right: The known occurrences of the Ariolimax Columbianus zoomed into California.

Figures 9 and 10 compare the predicted suitability for the Ariolimax Columbianus against observed data. As the species with the largest sample size, this model will provide us with the clearest picture of the efficacy of the MaxEnt model. We can see some strong correlations between the predicted and observed patterns, particularly in Washington and Oregon. The model also seems to correctly identify areas that are unsuitable for the species. It is in California where we see some instances of the MaxEnt model over-predicting. Highlighted in the red and blue circles in Figure 10, we see areas where the model has predicted high suitability, but no occurrences of the species have been recorded. This could have several causes, which we will analyse further as we continue.

This model achieved AUC score of 0.992, indicating a high degree of accuracy. AUC (area under the curve) quantifies how well the model distinguishes between suitable and unsuitable habitats [6]. A score of 1 represents a perfect classification, and 0.5 suggest the model is no better than random chance. While the AUC score can be influenced by sample size (an issue we will revisit later), the Columbianus model demonstrates excellent predictive performance.

All Individual Species' MaxEnt Models

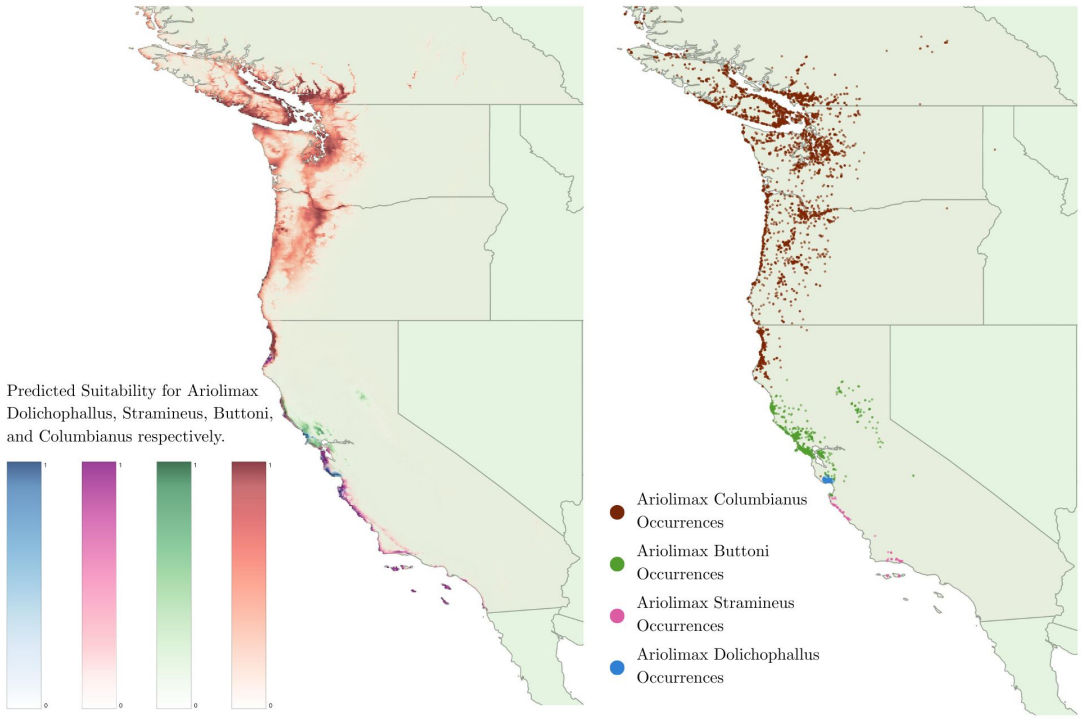

Figure 11: Left: Map of the MaxEnt models predicted suitability for each of the four species of Ariolimax over the research area. Right: Occurrences of the four species of Ariolimax over the research area.

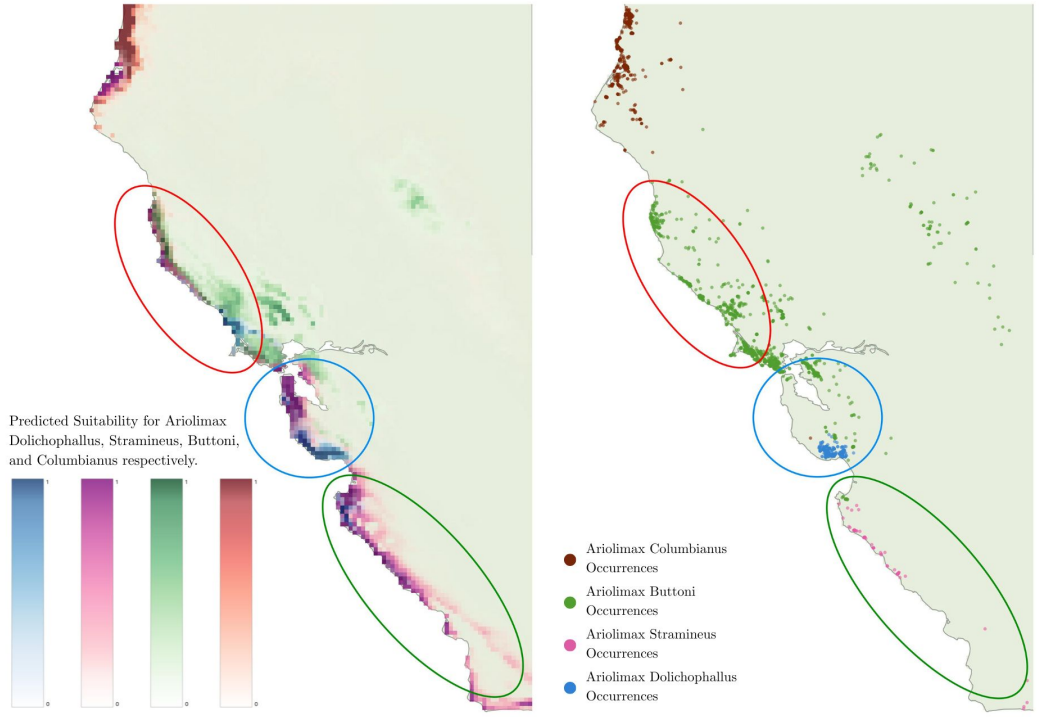

Figure 12: Left: Map of the MaxEnt models predicted suitability for each of the four species of Ariolimax zoomed into California. Right: the known occurrences of the four species of Ariolimax zoomed into California.

Figures 11 and 12 show MaxEnt models for all four Ariolimax species. Overall, we have strong correlations once again between the model predictions and the observed behaviour, though we see some differences emerge when we focus on California. In Figure 12, the red circle highlights where we see the Ariolimax Columbianus and Buttoni predicted to occur alongside each other, but only the Buttoni is observed. This suggests that we likely have some cross-species dynamics not captured by the model.

Similarly in the blue circle, the model has predicted the presence of both the Ariolimax Dolichophallus and Stramineus, whereas in reality there is no occurrence data at all. This may once again be due to cross-species dynamics, or an issue with the environmental rasters that we input into the model. Parts of this highlighted area are major cities: San Fransisco and San Jose, but there is also a large area of the Big Basin Redwoods National Park [8]. This may suggest that either the terrain and other environmental factors of this area is not suitable for the Ariolimax, or perhaps that the area is not suitable for the data collectors to observe the species. We also see another example of the model over-predicting, as seen in the green circle - a much higher predicted suitability of the Ariolimax Stramineus than the occurrence data would suggest.

We have already discussed the issues of cross-species dynamics, and so far the models have each been filtered and treated separately. We see that this can cause over-predictions, and thus we build a combined model, treating all Ariolimax species as a single dataset.

Combined Models against Separated Models

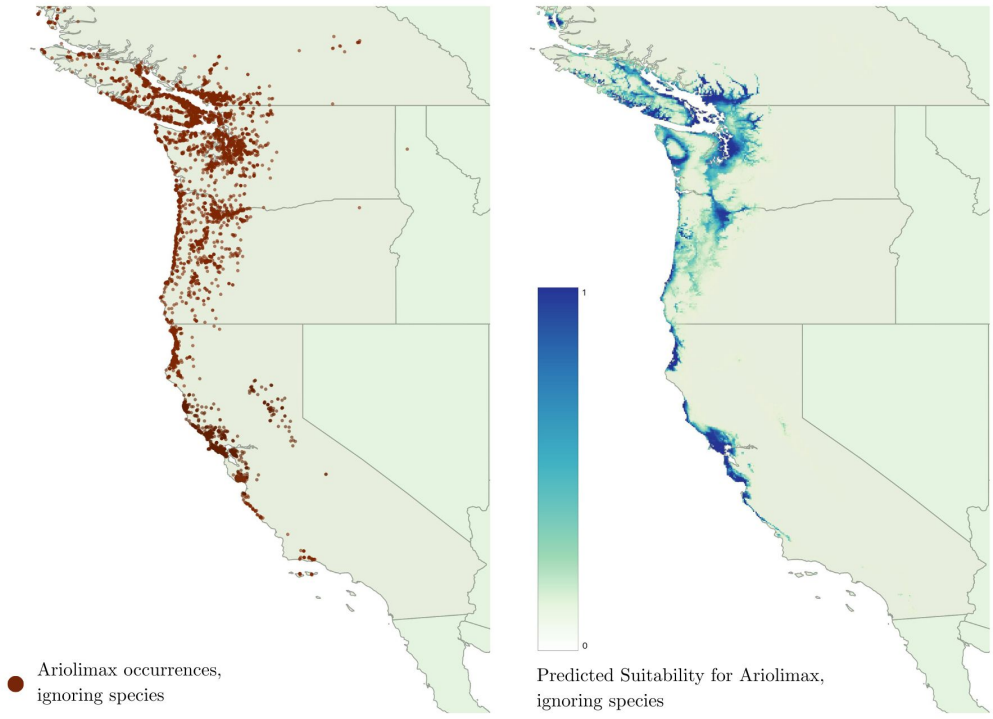

Figure 13: Left: Map of occurrences of any Ariolimax, ignoring species, across the research area. Right: MaxEnt model of predicted suitability of Ariolimax, ignoring species, across the research area.

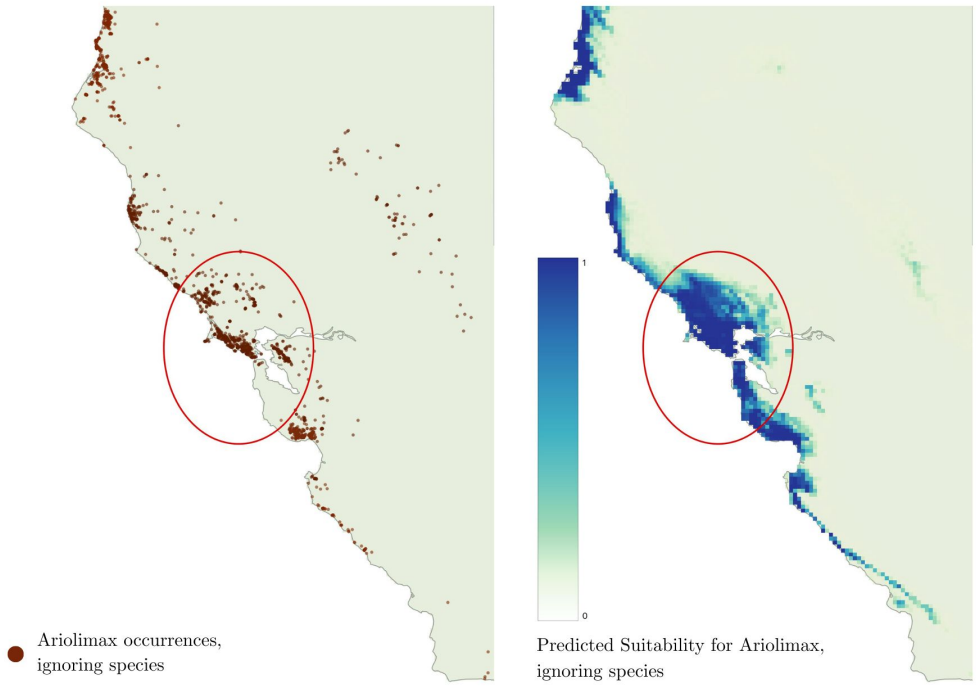

Figure 14: Left: Map of known locations of any Ariolimax occurrence, ignoring species, zoomed into California. Right: MaxEnt model of predicted suitability of Ariolimax, ignoring species, zoomed into California.

In Figures 13 and 14, we can see the occurrence data for all the Ariolimax locations, regardless of species, against a new MaxEnt model where the dataset was unfiltered and treated as a single dataset.The general spatial patterns remain consistent, and the major northern hotspots align well between data and predictions. It is again in California (Figure 14) where the over-predictions arise. The red circle shows a large area with high predicted suitability, but the occurrence data has some clear blank spots in the same location. Once again, there could be several reasons for this: cross-species competition, environmental factors that are ignored, such as canopy cover, not captured by temperature of precipitation alone.

As we can see in Figure 7, these areas of California have several species of Ariolimax in close quarters, and there could be competition for the basic necessities for the species to survive. There are some industrial areas in this location, but predominantly we are considering forested areas, which may indicated that some form of forest related data, like leaf cover or similar, should have been included in the input rasters. We also have an area north-east of the red circle which has a scattering of occurrences in the point data, but only a small indication of the species in the MaxEnt model, suggesting again that in order to improve the accuracy of the model, we may need to consider more environmental factors.

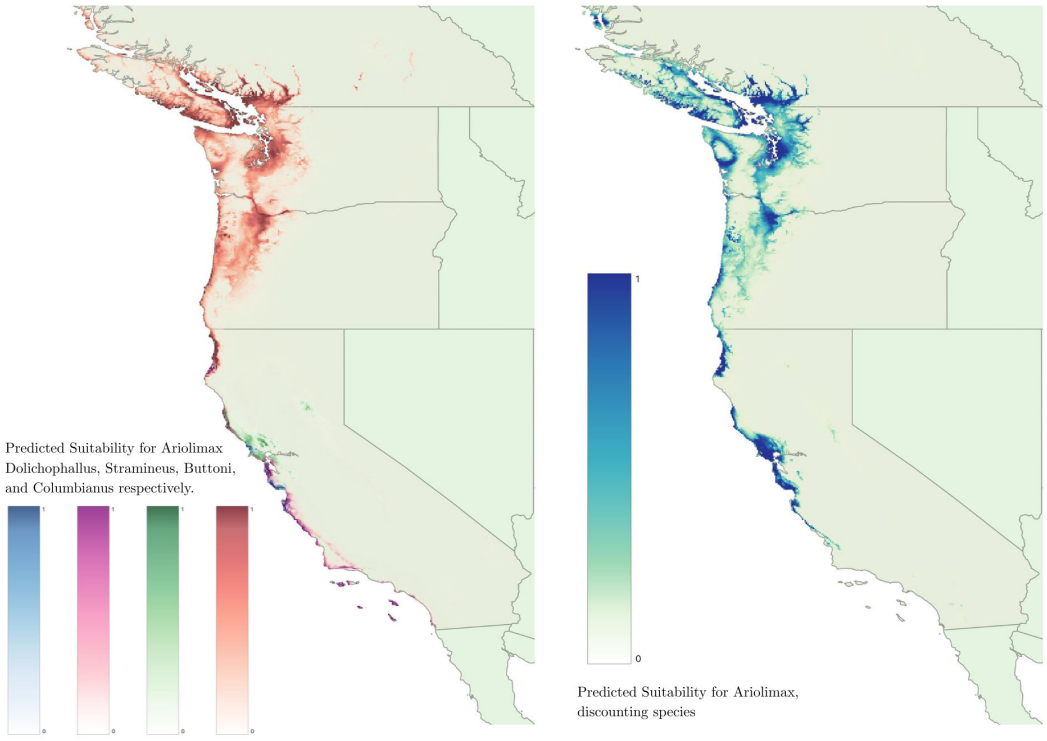

Figure 15: Left: Map of the MaxEnt models predicted suitability for each of the Ariolimax species. Right: MaxEnt model of predicted suitability of Ariolimax, ignoring species.

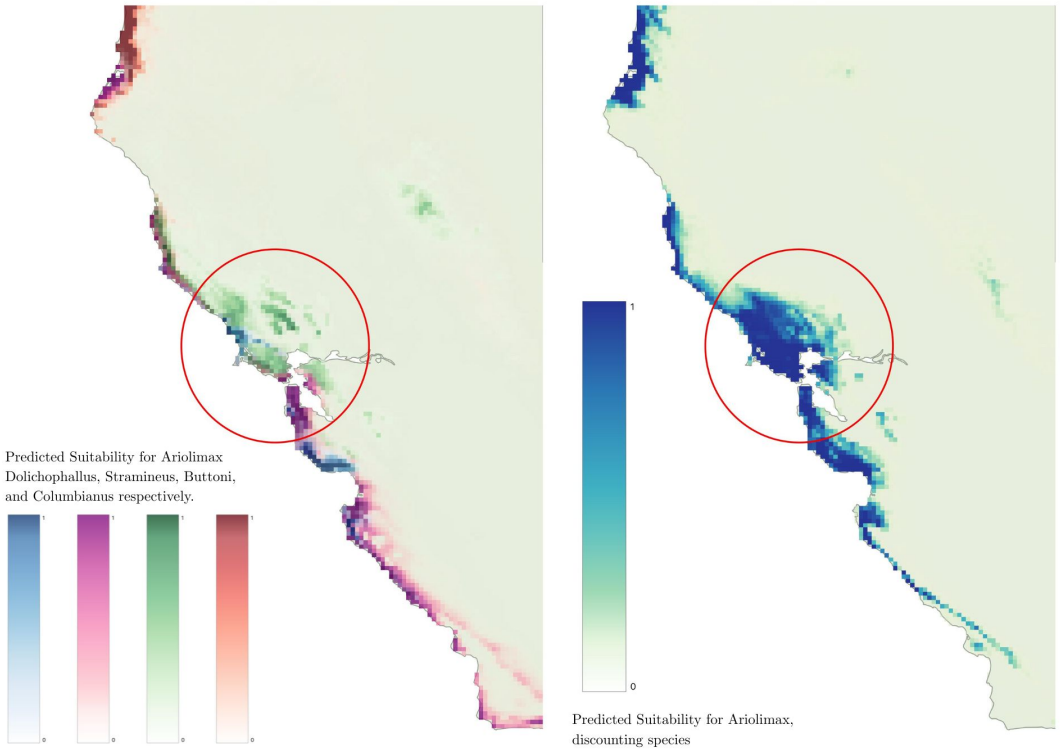

Figure 16: Left: Map of the MaxEnt models predicted suitability for each of the four species of Ariolimax zoomed into California. Right: MaxEnt model of predicted suitability of Ariolimax, ignoring species, zoomed into California.

Figures 15 and 16 directly compare the individual species' models with the combined model. Since both use identical environmental variables, we can assume that any difference arise from how the species interactions are represented. Once again, in Figure 14, we see the same over-prediction in the combined model on the right, and the same blank patches as the point data in the separate models on the left. This tells us that either the model doesn't fully incorporate cross species dynamics, or that the model's sampling process is affected by the separation of species. Comparing Figures 12 and 14, it is clear that separating the species yields a far more accurate model, an important consideration for further study.

Model Analysis

Having mapped the model results, we now assess their strengths and weaknesses to understand how the approach performs for SDMs, and where it could be improved.

Strengths

The Ariolimax Columbianus model closely matches observed distributions, which validates the MaxEnt approach for this species. The AUC scores further support this, especially in the Columbianus case of 0.992, and the Buttoni, 0.999. While models trained on smaller datasets tend to produce higher AUC scores due to over-fitting, the Columbianus result is particularly robust given its larger sample size. Overall, the MaxEnt method effectively distinguishes between suitable and unsuitable habitats.

Limitations and Future Improvements Several factors limit our model's accuracy. Firstly, we have the sample size of our occurrence data. Smaller datasets can artificially inflate AUC scores and reduce the models extrapolation and generalisability. Extending sampling coverage and ensuring data consistency would be the simplest way to improve the reliability of the model.

Secondly, we must consider the environmental predictors we trained the model on. While temperature and precipitation *are* key determinants, they are not the only factors that will influence the Ariolimax distribution. Incorporating additional rasters, such as vegetation cover, soil moisture, or topography would likely yield more ecologically realistic predictions.

Lastly, the cross-species dynamics. The MaxEnt framework treats each species independently and does not account for interactions between the species, such as competition or mutual exclusion. As seen in Figure 12, thus leads to over-predictions where multiple species are predicted to coexist but are not observed together, Developing models that explicitly incorporate inter-species interactions, or coupling MaxEnt with other new research areas like joint species distribution models, could address this limitation.

In summary, the MaxEnt approach demonstrates strong predictive power for the Ariolimax distributions, particularly for well-sampled species like the Columbianus. However, incorporating richer environmental data and modelling inter-species interactions are key directions for refining and extending this work.

6 Conclusion

Throughout this report, we have explored how ideas from entropy and convex optimisation come together to form the framework of maximum entropy modelling. We have seen these theories connect to applications, and how they underpin practical methods used in species distribution modelling. The MaxEnt model applied to the Ariolimax dataset produced results that closely reflected known distributions, particularly for species with larger datasets, such as the Columbianus. We have also identified the limitations of this particular model, in particular its inability to involve cross-species dynamics, an area that may provide insightful further research. Overall, the study shows how the mathematics of optimisation and entropy can provide meaningful insights into the systems that govern species distribution.

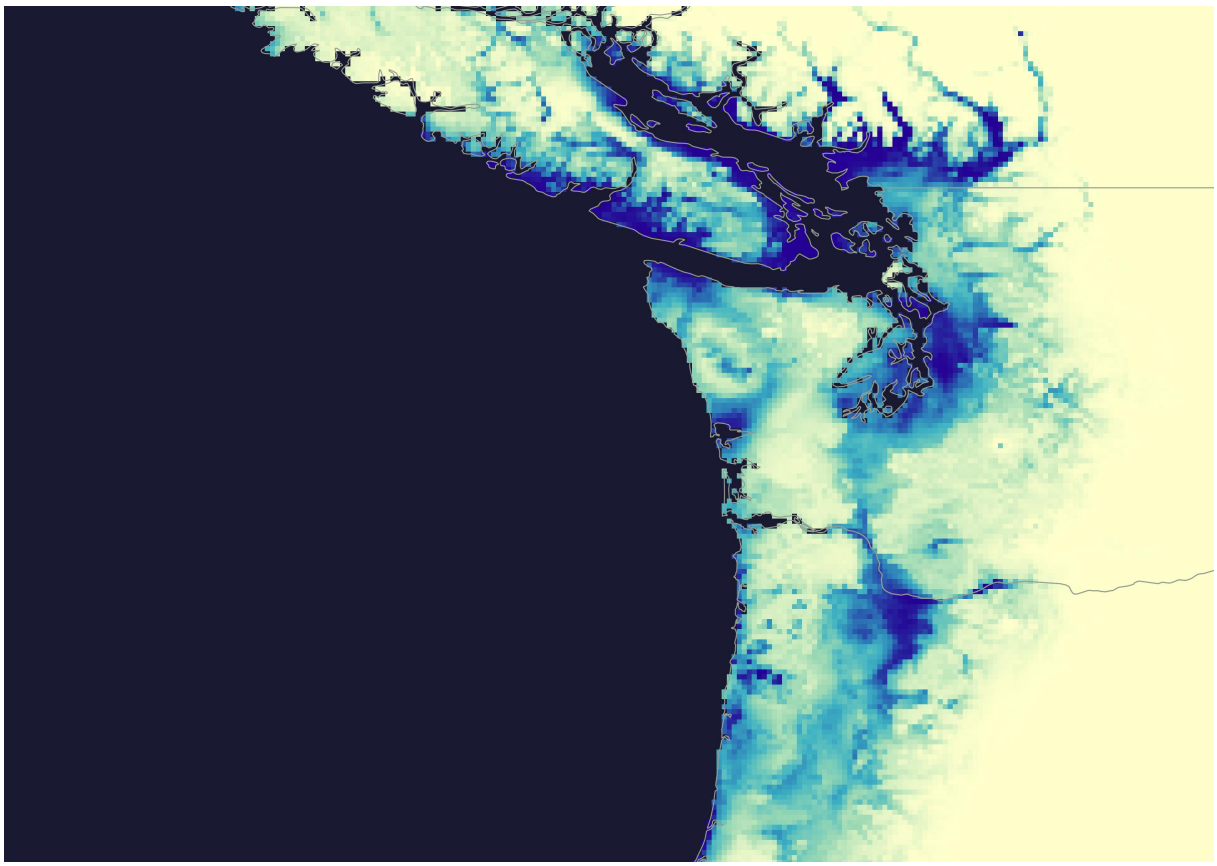

Figure 17: MaxEnt model of Ariolimax Columbianus

A Appendix: Code Sample

For full code for this project, refer to the repository in [9].

```
import os
import pandas as pd
import elapid as ela
from sklearn import metrics
import geopandas as gpd
import rasterio
from rasterio.windows import from_bounds

# paths
wc2_folder = "/wc2/clipped_research_area/"
locations_folder = "/filtered_locations_ariolimax/"
output_folder = "/ariolimax_rasters/"

os.makedirs(output_folder, exist_ok=True)

# raster data
raster_files = [os.path.join(wc2_folder, f) for f in os.listdir(wc2_folder) if
    f.endswith(".tif")]
background = ela.sample_raster(raster_files[0], count=10_000)

# bounding box
west, south, east, north = -140, 30, -110, 60

# store results
auc_scores = {}

# loop through all CSVs
for file in os.listdir(locations_folder):
    if file.endswith(".csv"):
        locations_file = os.path.join(locations_folder, file)

        # species name from filename
        species_name = file.replace("locations_", "").replace(".csv", "")
        print(f"\n--- Processing {species_name} ---")

        # load data
        locations = pd.read_csv(locations_file)
        species_all = locations.dropna(subset=['decimalLongitude', 'decimalLatitude'])

        geometry = ela.xy_to_geoseries(
            x=species_all['decimalLongitude'],
            y=species_all['decimalLatitude']
        )
        species_gdf = gpd.GeoDataFrame(
            species_all[['species', 'year']],
            geometry=geometry,
            crs="EPSG:4326"
        )

        print("Species records:", species_gdf.shape)

        # prep training data
        merged = ela.stack_geodataframes(species_gdf, background, add_class_label=True)
        merged = merged[merged.geometry.notnull() & merged.is_valid]
```

```
        annotated = ela.annotate(merged, raster_files, drop_na=True, quiet=True)

        x = annotated.drop(columns=['class', 'geometry'])
        y = annotated['class']

        # train model
        model = ela.MaxentModel(transform='cloglog', beta_multiplier=2.0)
        model.fit(x, y)

        # evaluation
        ypred = model.predict(x)
        auc = metrics.roc_auc_score(y, ypred)
        auc_scores[species_name] = auc
        print(f"AUC for {species_name}: {auc:.3f}")

        # prediction file
        output_file = os.path.join(output_folder,
            f"predicted_suitability_{species_name}.tif")
        ela.apply_model_to_rasters(model, raster_files, output_path=output_file)

        # clip
        clipped_file = os.path.join(output_folder,
            f"predicted_suitability_{species_name}_clipped.tif")
        with rasterio.open(output_file) as src:
            window = from_bounds(west, south, east, north, src.transform)
            clipped = src.read(1, window=window)
            transform = src.window_transform(window)
            meta = src.meta.copy()
            meta.update({
                "height": clipped.shape[0],
                "width": clipped.shape[1],
                "transform": transform
            })

        with rasterio.open(clipped_file, "w", **meta) as dst:
            dst.write(clipped, 1)

        print(f"Saved clipped raster: {clipped_file}")

# save AUC results to file
auc_file = os.path.join(output_folder, "auc_scores.txt")
with open(auc_file, "w") as f:
    for species, score in auc_scores.items():
        f.write(f"{species}: {score:.3f}\n")

print(f"\nAUC scores saved to {auc_file}")
```

List of Figures

References

[1] C E Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* (1948).

[2] Lieven Vandenberghe Stephen Boyd. *Convex Optimization*. Cambridge University Press, 2004.

[3] T Bossomaier; L Barnett; M Harré JT Lizier. *An introduction to transfer entropy : Information flow in complex systems*. Springer International Publishing AG, 2016.

[4] Baca-González V. Morales NS Fernández IC. "MaxEnt's parameter configuration and small samples: are we paying attention to recommendations? A systematic review." In: *PeerJ* (2017). DOI: doi:10.7717/peerj.3093. URL: https://doi.org/doi:10.7717/peerj.3093.

[5] Christopher B. Anderson. "elapid: Species distribution modeling tools for Python". In: *Journal of Open Source Software* (2023). DOI: 10.21105/joss.04930. URL: https://doi.org/10.21105/joss.04930.

[6] Aksel G. Çorbacıoğlu ŞK. "Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value." In: *Turkish journal of emergency medicine* (2023). DOI: 10.4103/tjem.tjem_182_23.

[7] gbif.org. *GBIF Occurence Download*. 2023. DOI: https://doi.org/10.15468/dl.f3zcby.

[8] Google. *Google Maps*. URL: https://www.google.com/maps.

[9] Ruby Keily-Thurstain. *Code Repository*. URL: https://github.com/ruby-kt/maxent_RP_2025/tree/main.

[10] Ben Stanfield. *Banana Slug Image*. URL: https://www.flickr.com/photos/acaben/516493964/.

[11] Lieven Vandenberghe Stephen Boyd. *Convex Optimisation Slides*. URL: https://web.stanford.edu/~boyd/cvxbook/bv_cvxslides.pdf. (accessed: Aug 2025).

[12] Animal Diversity Web. *Ariolimax columbianus*. URL: https://animaldiversity.org/accounts/Ariolimax_columbianus/. (accessed: Sept 2025).

[13] WorldClim. *Standard (19) WorldClim Bioclimatic Variables*. URL: https://www.worldclim.org/data/worldclim21.html. Resolution: 2.5 Minutes (accessed: Aug 2025).