Competence Development

Bayesian Decision Theory and Statistics Case Study

Jonas Petersen @ VELUX
Ander Runge Walther @ Energinet

Contents

1 Introduction

This study is going to consider the application of decision theory to generate statistically optimal decisions based on data. Decision theory is a field of study that provides a systematic framework for making decisions in the face of uncertainty. It is concerned with the process of choosing actions among several alternatives, taking into account the probabilities of different outcomes and the potential consequences associated with those outcomes. At its core, decision theory addresses the fundamental challenge of selecting the most optimal course of action when faced with uncertainty about the future. Decision-makers often confront situations where multiple choices are available, each with its associated risks and rewards. The goal is to make decisions that maximize expected utility or minimize expected loss, striking a balance between achieving favorable outcomes and mitigating potential risks.

A key concept in decision theory is the consideration of probabilities. Decision-makers must assess the likelihood of different outcomes and incorporate these probabilities into their decision-making process. This involves evaluating the uncertainty associated with each choice and weighing the potential gains against the potential losses.

Decision theory encompasses various decision-making models, each tailored to specific contexts and preferences. One common framework is the expected utility theory, which posits that decision-makers should choose actions that maximize the expected value of a utility function representing their preferences.

2 The Farmer Salesman

Consider a farmer who wishes to retire and therefore would like to sell their set of live animals. For simplicity, assume that animals make up a simple group and a given person can either be interested in purchasing an animal or not – the simplfication consist of not differentiating between different animal types. In order to sell their animals, the farmer needs to contact people with a sale in mind. For simplicity, the contact will be assumed to be via a telephone call only. The farmer can call (or not) with the intent to sell an animal to the recepient of the call. The recepient of the call can (or not) be interesed in purchasing an animal (Natures decision). Let $\mathbb{U}$ denote the set of the farmers actions and $\mathbb{S}$ the set of Natures actions, then

$$\begin{aligned} \mathbb{U} &= \{u_1 = \text{call}, u_2 = \text{don't call}\}, \\ \mathbb{S} &= \{s_1 = \text{interested}, s_2 = \text{not interested}\}. \end{aligned} \tag{1}$$

A nuisance for the contacted people is associated to the call which is represented by the abstract monetary loss, $\lambda \in \mathbb{R}^+$. The degree to which people are annoyed by a sales call is independent in general and the monetary loss represents the average animosity generated and the associated moneraty loss connected to a worsened reputation. Aside from the nuisance associated with a sales call, there is also a moneraty reward for a successfull sale, ψ. If the farmer cannot sell his animals, he will have them terminated with no associated cost in order not to spend additional

time or money on them. Given these assumptions, the cost function can be represented by the matrix

		S	
		$s_1 =$ Interested	$s_2 =$ Not interested
U	$u_1 =$ Call	$\lambda - \psi$	λ
	$u_2 =$ Don't call	0	0

The farmer has available to them observations $X = x$ that contain information regarding the decesion Nature is going to make, $S = s$. The farmer also have a collection of past observations and resulting decisions of Nature, i.e. $D = \{(X = x_1, S = s_1), (X = x_2, S = s_2), \dots (X = x_n, S = s_n)\} = D_s \times D_s$. The optimal decision for the farmer to call the i'th person can then be written viz

$$U^*(x) = \arg\min_{U(x)} \mathbb{E}_{S|X}[C(U(x), S)|X = x, D, I], \tag{2}$$

where

$$\mathbb{E}_{S|X}[C(U(x), S)|X = x, D, I] = \sum_{s \in \mathbb{S}} C(U(x), s) p(S = s|X = x, D, I). \tag{3}$$

Writing out the conditional expectation

$$\begin{aligned}
\mathbb{E}_{S|X}[C(u_1, S)|x, D, I] &= \sum_s C(u_1, s) p(S = s|x, D, I) \\
&= C(u_1, s_1) p(S = s_1|x, D, I) + C(u_1, s_2) p(S = s_2|x, D, I) \\
&= (\lambda - \psi) p(S = s_1|x, D, I) + \lambda_i p(S = s_2|x, D, I) \\
\mathbb{E}_{S|X}[C(u_2, S)|x, D, I] &= \sum_s C(u_2, s) p(S = s|x, D, I) \\
&= C(u_2, s_1) p(S = s_1|x, D, I) + C(u_2, s_2) p(S = s_2|x, D, I) \\
&= 0
\end{aligned} \tag{4}$$

The optimal decision rule $U^*(x)$ can be implicitly specified as picking u_1 (call) iff $\mathbb{E}_{S|X}[C(u_1, S)|x, D, I] < \mathbb{E}_{S|X}[C(u_2, S)|x, D, I]$, corresponding to picking u_1 (call) iff

$$(\lambda - \psi) p(S = s_1|x, D, I) + \lambda p(S = s_2|x, D, I) < 0 \tag{5}$$

Since $p(S = s_1|x, D, I) + p(S = s_2|x, D, I) = 1$

$$\frac{\lambda}{\psi} < p(S = s_1|x, D, I) \tag{6}$$

meaning the farmer should call (action u_1) iff the probability for the recepient of the call to be interested in at least one animal is larger than the penalty of calling divided by the gain of calling.

2.1 The Probability

Equation (6) implicitly specify the decision rule for the farmer. λ, ψ is assumed specified, so only the probability $p(S = s_1|x, D, I)$ remain to be specified. Using marginalization and assuming independence

$$\begin{aligned}
p(S = s|x, D, I) &= \int p(S = s, \theta|x, D, I) d\theta \\
&= \int p(S = s|x, \theta, D, I) p(\theta|x, D, I) d\theta \\
&= \int p(S = s|x, \theta, I) p(\theta|D, I) d\theta.
\end{aligned} \tag{7}$$

Suppose now a model, $f : \mathbb{W} \times X \mapsto [0, 1]$, with associated parameters $\theta \in \mathbb{W}$, that estimates Nature's actions S based on observed data X is introduced. The random variable S is discrete and the function is identified with the probability of each action

$$p(S = s|x, \theta, I) = f_{S=s}(\theta, x), \tag{8}$$

with

$$\sum_{s \in \mathbb{S}} p(S = s|x, \theta, I) = 1. \tag{9}$$

Combining equation (7) and (8)

$$\begin{aligned} p(S = s|x, D, I) &= \int f_{S=s}(\theta, x) p(\theta|D, I) d\theta \\ &= \mathbb{E}[f_{S=s}(\theta, x)|D, I]. \end{aligned} \tag{10}$$

From Bayes theorem

$$p(\theta|D, I) = \frac{p(D_s|D_x, \theta, I) p(\theta|D_x, I)}{p(D_s|D_x, I)}, \tag{11}$$

where $p(\theta|D_x, I) = p(\theta|I)$. Assuming the distribution over θ is normally distributed with zero mean and a precision described by a hyperparameter, λ,

$$p(\theta|I) = \int p(\theta|\lambda, I) p(\lambda|I) d\lambda. \tag{12}$$

The precision is constructed as a wide gamma distribution so as to approximate an objective prior

$$p(\theta|\lambda, I) p(\lambda|I) = \prod_{q=1}^{\tilde{n}} \frac{\lambda_q^{\frac{n_q}{2}}}{(2\pi)^{\frac{n_q}{2}}} e^{-\frac{\lambda_q}{2} \sum_{l=1}^{n_q} \theta_l^2} \frac{\beta_q^{\alpha_q}}{\Gamma(\alpha_q)} \lambda_q^{\alpha_q - 1} e^{-\beta_q \lambda_q} \tag{13}$$

Assuming the past actions of Nature are independent and identically distributed, the likelihood can be written

$$\begin{aligned} p(D_s|D_x, \theta, I) &= \prod_{i=1}^{n} p(S = s_i|X = x_i, \theta, I) \\ &= \prod_{i=1}^{n} f_{s_i}(\theta, x_i) \end{aligned} . \tag{14}$$

Aside from the specification of the model f, $p(S = s|x, D, I)$ is at this point fully specified and can be approximated by HMC similarly to the regression case. In this case, the model can be represented by the Hamiltonian

$$H \equiv \sum_q \sum_l \frac{p_l^2}{2m_l} - \ln(p(\theta, \lambda|D, I)) + const \tag{15}$$

where

$$p(\theta|D, I) = \int d\lambda p(\theta, \lambda|D, I). \tag{16}$$

Using equations (7)-(14) in equation (15) yields the Hamiltonian

$$\begin{aligned} H = &\sum_{q=1}^{\tilde{n}} \sum_{l=1}^{n_q} \frac{p_l^2}{2m_l} - \sum_{i=1}^{n} \ln(f_{s_i}(\theta, x_i)) + \text{const} \\ &+ \sum_{q=1}^{\tilde{n}} \Bigg(\ln(\Gamma(\alpha_q)) - \alpha_q \ln(\beta_q) + (1-\alpha_q)\ln(\lambda_q) + \beta_q \lambda_q. \\ &\qquad + \frac{n_q}{2}(\ln(2\pi) - \ln(\lambda_q)) + \frac{\lambda_q}{2} \sum_{l=1}^{n_q} \theta_l^2 \Bigg) \end{aligned} \tag{17}$$

2.2 Simple Model

Let

$$f_{S=s}(\theta, x_i) = \frac{e^{b_s + \sum_q a_{sq} x_{iq}}}{\sum_{k \in \mathbb{S}} e^{b_k + \sum_q a_{kq} x_{iq}}}, \tag{18}$$

where $\theta = \{b, a\}$.

2.3 Manual HMC Algorithm

The Hamiltonian is given by

$$\begin{aligned} H = &\sum_{q=1}^{2} \sum_{l=1}^{2} \frac{p_{ql}^2}{2m_{ql}} - \sum_{i=1}^{n} \ln(f_{s_i}(\theta, x_i)) \\ &+ \ln(\Gamma(\alpha_a)) - \alpha_a \ln(\beta_a) + (1-\alpha_a)\ln(\lambda_a) + \beta_a \lambda_a \\ &\qquad + \frac{1}{2}(\ln(2\pi) - \ln(\lambda_a)) + \frac{\lambda_a}{2} \sum_{j,q} a_{jq}^2 \\ &+ \ln(\Gamma(\alpha_b)) - \alpha_b \ln(\beta_b) + (1-\alpha_b)\ln(\lambda_b) + \beta_b \lambda_b \\ &\qquad + \frac{1}{2}(\ln(2\pi) - \ln(\lambda_b)) + \frac{\lambda_b}{2} \sum_{j} b_j^2 \end{aligned} \tag{19}$$

λ_j is positive definite. In order to uphold this numerically, let $\lambda_j = e^{\tau_j}$. When making this transformation, the integration measure of equation (12) has to be transformed as well. This proceeds viz

$$d\lambda_j = \lambda_j d\tau_j, \tag{20}$$

meaning effectively λ_j is multiplied on $p(\theta, \lambda | D, I)$ such that $H \to H - \ln(\lambda_j)$. This means

$$(1-\alpha_j)\ln(\lambda_j) \in H \Rightarrow -\alpha_j \ln(\lambda_j). \tag{21}$$

Additionally, it is convenielt to pick out the s_i via a one-hot target vector such that

$$\begin{aligned} H = &\sum_{q=1}^{2}\sum_{l=1}^{2}\frac{p_{ql}^2}{2m_{ql}} - \sum_{j\in\mathbb{S}}\sum_{i=1}^{n} s_{ij}\ln(f_j(\theta, x_i)) \\ &+ \ln(\Gamma(\alpha_a)) - \alpha_a\ln(\beta_a) - \alpha_a\tau_a + \beta_a e^{\tau_a} \\ &\quad + \frac{1}{2}(\ln(2\pi) - \tau_a) + \frac{e^{\tau_a}}{2}\sum_{j,q} a_{jq}^2 \\ &+ \ln(\Gamma(\alpha_b)) - \alpha_b\ln(\beta_b) - \alpha_b\tau_b + \beta_b e^{\tau_b} \\ &\quad + \frac{1}{2}(\ln(2\pi) - \tau_b) + \frac{e^{\tau_b}}{2}\sum_{j} b_j^2 \end{aligned} \tag{22}$$

The derivatives are needed for the HMC algorithm

$$\frac{\partial H}{\partial a_{ml}} = -\sum_{i,j}\frac{s_{ij}}{f_{ij}}\frac{\partial f_{ij}}{\partial a_{ml}} + e^{\tau_a}a_{ml}, \tag{23}$$

$$\begin{aligned} \frac{\partial f_{ij}}{\partial a_{ml}} = &\frac{e^{b_j+\sum_{q_1} a_{jq_1}x_{iq_1}}}{\sum_{k\in\mathbb{S}} e^{b_k+\sum_{q_2} a_{kq_2}x_{iq_2}}}\sum_{q_3}\delta_{jm}\delta_{q_3 l}x_{iq_3} \\ &- \frac{e^{b_j+\sum_{q_4} a_{jq_4}x_{iq_4}}}{(\sum_{k\in\mathbb{S}} e^{b_k+\sum_{q_5} a_{kq_5}x_{iq_5}})^2}\sum_{k'\in\mathbb{S}} e^{b_{k'}+\sum_{q_6} a_{k'q_6}x_{iq_6}}\sum_{q_7}\delta_{k'm}\delta_{q_7 l}x_{iq_7} \\ = &f_{ij}\delta_{jm}x_{il} - f_{ij}f_{im}x_{il} \end{aligned} \tag{24}$$

where it has been used that

$$\frac{\partial a_{jq_3}}{\partial a_{ml}} = \delta_{jm}\delta_{q_3 l} \tag{25}$$

$$\begin{aligned} \frac{\partial H}{\partial a_{ml}} &= -\sum_{i,j}\frac{s_{ij}}{f_{ij}}(f_{ij}\delta_{jm}x_{il} - f_{ij}f_{im}x_{il}) + e^{\tau_a}a_{ml} \\ &= \sum_{i} x_{il}(f_{im} - s_{im}) + e^{\tau_a}a_m \end{aligned} \tag{26}$$

$$\frac{\partial H}{\partial b_m} = \sum_{i}(f_{im} - s_{im}) + e^{\tau_b}b_m. \tag{27}$$

$$\frac{\partial H}{\partial \tau_m} = -\alpha_m + \beta_m e^{\tau_m} - \frac{1}{2} + \frac{e^{\tau_m}}{2}\sum_{j} m_j^2. \tag{28}$$

The masses for the HMC algorithm can be set by approximating the second order derivatives as fixed. Let

$$\begin{aligned} \frac{\partial^2 H}{\partial a_{ml}^2} &= \sum_{i} x_{il}\frac{\partial f_{im}}{\partial a_{ml}} + e^{\tau_a} \\ &= \sum_{i} x_{il}^2 f_{im}(1 - f_{im}) + e^{\tau_a} \end{aligned} \tag{29}$$

then taking $x_{il}^2 \sim 1, f_{im} \sim \frac{1}{2}$ and any parameter ~ 0, meaning $e^{\tau_a} \sim 1$ yield the mass approximation

$$\begin{aligned} m_{ml}^{(a)} &\sim \left.\frac{\partial^2 H}{\partial a_{ml}^2}\right|_{\text{fixed approximation}} \\ &\sim N \cdot 1^2 \cdot \frac{1}{2}(1-\frac{1}{2}) + 1 \\ &= \frac{N}{4} + 1, \end{aligned} \tag{30}$$

where N is the number of data samples in D_x. Similarly

$$\begin{aligned} \frac{\partial^2 H}{\partial b_m^2} &= \sum_i \frac{\partial f_{im}}{\partial a_{ml}} + e^{\tau_b} \\ &= \sum_i f_{im}(1-f_{im}) + e^{\tau_b}, \end{aligned} \tag{31}$$

meaning (since $x_{il}^2 \sim 1$)

$$\begin{aligned} m_{ml}^{(a)} &\sim \left.\frac{\partial^2 H}{\partial b_m^2}\right|_{\text{fixed approximation}} \\ & m_{ml}^{(b)}. \end{aligned} \tag{32}$$

The precision parameter

$$\frac{\partial^2 H}{\partial \tau_q^2} = \beta_q e^{\tau_q} + \frac{e^{\tau_q}}{2} \sum_j q_j^2. \tag{33}$$

Take $\beta_q = 3$, then

$$\begin{aligned} m_q^{(\tau)} &\sim \left.\frac{\partial^2 H}{\partial \tau_q^2}\right|_{\text{fixed approximation}} \\ &\sim 3. \end{aligned} \tag{34}$$

2.4 Data

Take $x = (\text{area}, \text{number of animals})^T$ and $s = 2\text{d}$ one-hot vector, with $\dim(D) = 1000$. D is split into two sets $D^{(\text{training})}$ and $D^{(\text{test})}$, with $\dim(D^{(\text{training})}) = \gamma \dim(D)$ and $\dim(D^{(\text{test})}) = (1-\gamma)\dim(D)$ and $\gamma = 0.6$. $D^{(\text{training})}$ will be used to train the model and $D^{(\text{test})}$ to evaluate the quality of the trained model. The underlying truth of Nature (unbeknownst to the model) is that an animal will be purchased iff

$$\text{total area} - 3.2 \cdot \text{number of animals} \geq 3.2. \tag{35}$$

2.5 Training

Using $D^{(\text{training})}$ as input, the algorithms the algorithms are trained for 2000 iterations. The first 500 iterations are taken as burn in to be conservative. The coefficients, θ, for iterations $[500, 2000]$ are used to make a model prediction viz

$$p(S = s|x, D, I) = \frac{1}{1500} \sum_{i=500}^{2000} f(\theta_i, x) \tag{36}$$

The accuracy of the modeled probabilities can be gauged by considering the case where $\psi = 2\lambda$ such that the decision rule (equation (6)) becomes

$$\frac{1}{2} < p(S = s_1|x, D, I), \tag{37}$$

and the classification is driven by the probabilities alone.

2.6 PyMC HMC Algorithm

PyMC is a probabilistic programming library for Python that allows users to build Bayesian models with a Python API and fit them using Markov Chain Monte Carlo methods PyMC link. Using this API it is possible to create, train and test a PyMC-equivalent of the model described in the previous sections. The general approach to building a model using PyMC consists of stating the data generating process, specifying a likelihood, and related prior distributions for any parameters involved. While modeling the data generating process and framing the statistical problem correctly are never completely trivial and require some effort from the user it is rather straight forward to perform the Markov Chain Monte Carlo sampling with PyMC. The user do not need to do any calculations related to the Hamiltonian Monte Carlo method nor do they need to specifically handle any integrals. In addition, there is a range of predefined probability distributions both discrete and continuous readily available in the library such as the Gamma (figure 1) and Normal distribution used for modeling the priors as described in section 2.1.

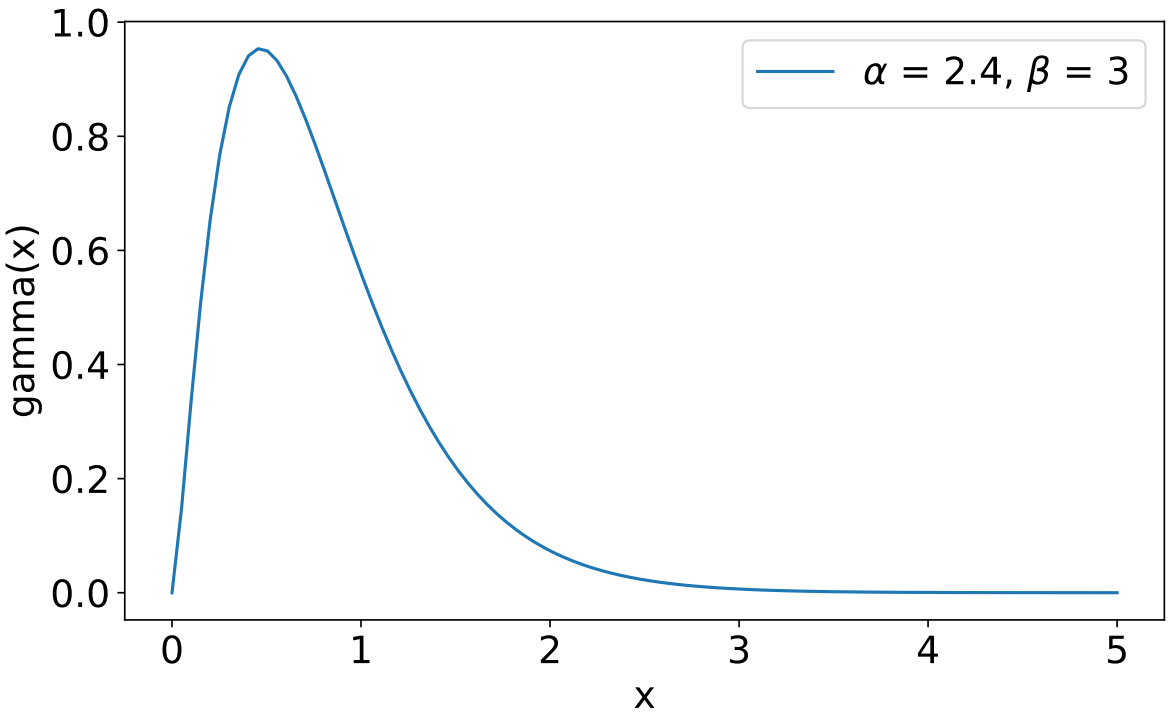

Figure 1: Plot of Gamma probability distribution used as priors for the precision parameters in Normal distributions.

The python code in algorithm 2.6 shows how the simple model can be declared using PyMC. All models in PyMC are declared using a "with pm.Model() as modelname"-statement, where pm is the abbreviated form of the PyMC package. Within this statement at range of random

variables, data and their relations can be declared. The syntax for declaring a random variable by its probability distribution (such as pm.Normal) is to pass a name for the random variable as a string for the first argument. The rest of the arguments are typically values for the parameters specific to the distribution. PyMC random variables can often be used for stating parameters in other random variables forming a hierarchy of distributions as is the case for the precision variables used to model the variance for the normally distributed parameters a and b (see equation 18) with zero mean. In the code example, these parameters are then combined deterministically with D_x to constitute the data generating process of the simple model. D_x is declared as a PyMC mutable object so that it will be possible later to change the values to generate predictions. As a last step the likelihood is declared. In PyMC this is the step where the relationship between the observed conditional (D_s) data and parameters is stated. As the output is the probabilities for the two classes a Multinomial distribution with the number of independent trials (n) equal to 1 is used as the likelihood.

Algorithm 1 PyMC Python Code

```
import pymc as pm
import numpy as np
with pm.Model() as classifier_model:
covars = pm.MutableData('covars',data_x_training)
# Priors for precision
precision_a = pm.Gamma('precision_a', alpha=2.4, beta=3)
precision_b = pm.Gamma('precision_b', alpha=2.4, beta=3)
# Priors for parameters
param_a = pm.Normal('param_a',
0,
sigma=1/np.sqrt(precision_a),
shape=(2,2))
param_b = pm.Normal('param_b',
0,
sigma=1/np.sqrt(precision_b),
shape=(2,))
# Data generating process
T = pm.Deterministic('T',pm.Math.exp(param_b + pm.math.dot(covars,
                                    param_a.T)))
class_conditional_probability = pm.Deterministic('
                                    class_conditional_probability', T/T.sum
                                    (axis=1, keepdims=True))
# Likelihood
obs = pm.Multinomial('obs', n=1, p=class_conditional_probability,
                                    observed=data_s_training, shape=
                                    class_conditional_probability.shape)
```

After declaring the model it is possible to sample the posterior by calling the pm.sample() method. Burn-in can be controlled by setting the tune argument. The draw argument determines how many samples are being drawn while a number of chains can be run in parallel by setting the chains and cores (computational) arguments.

Algorithm 2 PyMC Posterior Python Code

```
with classifier_model:
posterior = pm.sample(tune=512, draws=1024, chains=4, cores=4)
```

Using the result of drawing samples from the distribution of model parameters (posterior) it is possible to draw from the posterior predictive distribution using the pm.sample_posterior_predictive() method. Without changing the input data this is equivalent to obtaining the result of applying the trained model on $D_x^{(\text{training})}$. By changing the input data to $D_x^{(\text{test})}$ using the pm.set_data() method it is possible to obtain the posterior predictive distribution.

Algorithm 3 PyMC Posterior Python Code

```
with classifier_model:
posterior_predictive = pm.sample_posterior_predictive(posterior)
pm.set_data({'covars': data_x_test})
posterior_predictive_test = pm.sample_posterior_predictive(posterior)
```

Using 37 for $D^{(\text{training})}$, the PyMC model correctly classify 598 of 600 data points. For $D^{(\text{test})}$, the model correctly classify 398.

2.7 Results

2.7.1 Manual HMC Algorithm

The HMC algorithm have parameters "step_scale" and "number_of_steps_scale", which adjust the overall scale of the step lengths and number of steps in pase space. Ideally, the distance between points should be large, so that step scale should be small (what the step length is divided by should be small) and the number of steps should be large. Numerical stability only exist for step_scale$\gtrsim 5$ (given accurate mass estimation) and thus only the number of steps remain as a variable to tune. In this study step_scale$= 10$ and number_of_steps_scale$= 1500$, where the latter is limited by reasonable computation time (to match approximately the pymc computation time). Given these parameters, the manual HMC algorithm misclassify a single training data point

$$x^{(\text{training})}_{\text{misclassified 1}} = \begin{pmatrix} 3.21798365 \\ 0 \end{pmatrix} \tag{38}$$

and two test data points

$$x^{(\text{test})}_{\text{misclassified 1}} = \begin{pmatrix} 6.40501793 \\ 1 \end{pmatrix}, \quad x^{(\text{test})}_{\text{misclassified 2}} = \begin{pmatrix} 9.60627827 \\ 2 \end{pmatrix}. \tag{39}$$

With equation (35) in mind, it is clear that the misclassifications of equation (38) and (39) are close to the limit with respect to purchasing an animal.

2.7.2 PyMC HMC Algorithm

The PyMC HMC Algorithm obtain misclassify two training data points; equation (38) and

$$x^{(\text{training})}_{\text{misclassified 2}} = \begin{pmatrix} 12.80826256 \\ 3 \end{pmatrix} \tag{40}$$

and two test data points (equation (39)).

3 Summary and Discussion

It has been shown how decision theory can be used in conjunction with statistics to make theoretically optimal decisions based on a user specified set of preferences (cost function). Using mock data, a "manual HMC algorithm" written by hand and a standard Python "PyMC HMC algorithm" have been compared. The two yield identical results on test data with the manual model yielding marginally better results on training data. Overall the performace is deemed equivalent both in terms of accuracy and computational speed. The manual HMC algorithm require the user to derive the gradients, write the sampling algorithm in Python and tune the algorith, whereas the latter only require a specification of the model via a standardized PyMC interface. Hence, from a user complexity perspective, the PyMC algorithm has a significant advantage.