1 The Blackjack Problem I: Controlling Player Decision

We define a hand of blackjack as complete at time k, where the player's cards are dealt at times $\tau_k = 1, \ldots, T_k^{(p)}$ and recorded as the vector $\mathbf{c}_k^{(p)} = \left[c_{k,1}^{(p)}, \ldots, c_{k,T_k^{(p)}}^{(p)}\right]' \in \{1, 2 \ldots, 10\}^{T_k^{(p)}}$ - an Ace is encoded as 1 (not to be confused with its potential hand-value of either 1 or 11), face cards (J, Q, and K) are encoded as 10, and all other cards retain their nominal face values. The dealer's single upcard is denoted by the scalar $c_k^{(d,\text{up})}$. The dealer's hole card (revealed only after the player's turn ends) at time $\tau_k = T_k^{(p)} + 1$ is denoted by the scalar $c_k^{(d,\text{hole})}$, and any additional dealer cards drawn at times $\tau_k = T_k^{(p)} + 2, \ldots, T_k$ are recorded as the vector $\mathbf{c}_k^{(d)} = \left[c_{k,T_k^{(p)}+2}^{(d)}, \ldots, c_{k,T_k}^{(d)}\right]' \in \{1, 2 \ldots, 10\}^{T_k - T_k^{(p)} - 1}$.

We treat both the player's and the dealer's turns as two ordered sequences of events, and define the card history for the k^{th} blackjack hand as $\mathcal{H}_k = \left[\left(\mathbf{c}_k^{(p)}\right)', c_k^{(d,\text{up})}, c_k^{(d,\text{hole})}, \left(\mathbf{c}_k^{(d)}\right)'\right]' \in \{1, 2 \ldots, 10\}^{T_k}$ where $\mathcal{H}_k$ represents all cards eventually observed by the player in the k^{th} hand. Furthermore, we define the complete card history observed by the player at the end of the k^{th} (from the 1^{st} hand of blackjack) as $\boldsymbol{\mathcal{H}}_k = \left[\mathcal{H}_1', \mathcal{H}_2', \ldots, \mathcal{H}_k'\right]'$. Additionally, we define the complete set of the dealer's cards observed in the k^{th} hand as $\mathbf{c}_k^{(d,\text{all})} = \left[c_k^{(d,\text{up})}, c_k^{(d,\text{hole})}, \left(\mathbf{c}_k^{(d)}\right)'\right]' \in \{1, 2 \ldots, 10\}^{T_k - T_k^{(p)}}$. Furthermore, we represent the partial player hand during the k^{th} blackjack hand at time τ_k, where $\tau_k \leq T_k^{(p)}$, as $\mathbf{c}_{k,1:\tau_k}^{(p)} = \left[c_{k,1}^{(p)}, \ldots, c_{k,\tau_k}^{(p)}\right]' \in \{1, 2, \ldots, 10\}^{\tau_k}$. The corresponding partial card state including the dealer's upcard is then given by $\boldsymbol{c}_{k,\tau_k}^{\text{partial}} = \left[\left(\mathbf{c}_{k,1:\tau_k}^{(p)}\right)', c_k^{(d,\text{up})}\right]' \in \{1, 2, \ldots, 10\}^{\tau_k + 1}$.

1.1 The hand's outcome

We define the hand-value of a player's or dealer's cards, $|\mathbf{c}| = R$ as such:

$$h(\mathbf{c}) = \begin{cases} \sum_{i=1}^{R} c_i + 10, & \text{if } c_i = 1 \text{ and } \sum_{i=1}^{R} c_i + 10 \leq 21, \\ \sum_{i=1}^{R} c_i & \text{otherwise.} \end{cases}$$

which implies that if an Ace is present ($c_i = 1$) and the total hand-value does not exceed 21 when the Ace is treated as 11, then the Ace is valued at 11. Otherwise, it is valued as 1. We record the bet multiplier at the conclusion of the k^{th} hand, s_k, as such:

$$s_k = \begin{cases} +2, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) > h\left(\mathbf{c}_k^{(d,\text{all})}\right) \text{ and } h\left(\mathbf{c_k}^{(p)}\right) \leq 21 \text{ and player chose to double-down where } |\mathbf{c_k}^{(p)}| = 3, \\ +2, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) \leq 21 \text{ and } h\left(\mathbf{c}_k^{(d,\text{all})}\right) > 21 \text{ and player chose to double-down,} \\ +1.5, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) = 21 \text{ for } |\mathbf{c_k}^{(p)}| = 2 \text{ and } h\left(\left[c_k^{(d,\text{up})}, c_k^{(d,\text{hole})}\right]'\right) \neq 21, \\ +1, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) > h\left(\mathbf{c}_k^{(d,\text{all})}\right) \text{ and } h\left(\mathbf{c_k}^{(p)}\right) \leq 21 \text{ and player did not choose to double-down,} \\ +1, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) \leq 21 \text{ and } h\left(\mathbf{c}_k^{(d,\text{all})}\right) > 21 \text{ and player did not choose to double-down,} \\ -2, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) > 21 \text{ and player chose to double-down where } |\mathbf{c_k}^{(p)}| = 3, \\ -1, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) > 21 \text{ and player did not choose to double-down,} \\ -1, & \text{if } h\left(\mathbf{c_k}^{(p)}\right) < h\left(\mathbf{c}_k^{(d,\text{all})}\right) \text{ and } h\left(\mathbf{c_k}^{(d,\text{all})}\right) \leq 21, \\ -0.5, & \text{if player surrendered hand,} \\ 0, & \text{otherwise.} \end{cases}$$

1.2 Control: Player decision

We control player decisions/actions at time $\tau_k = 1, \ldots, T_k^{(p)}$ for the k^{th} hand of blackjack through the means of the control vector $\mathbf{ct}\left(\boldsymbol{c}_{k,\tau_k-1}^{\text{partial}}, \boldsymbol{\theta}^{\text{Decision}}\right) \in \mathcal{A}_{\tau_k} \subseteq \{\text{Stay}, \text{Hit}, \text{Split}, \text{Surrender}, \text{Double-Down}\}$ for some parameter configuration $\boldsymbol{\theta}^{\text{Decision}} \in \mathbb{R}^R$. We note that $\mathcal{A}_{\tau_k}$ denotes the subset of admissible actions available at time $\tau_k \leq T_k^{(p)}$, immediately prior to the player's decision. Splitting is allowed only if the player's first two cards are identical $c_{k,1}^{(p)} = c_{k,2}^{(p)}$ and can only occur at $\tau_k = 2$, and surrendering and doubling down can only occur at $\tau_k = 2$. Moreover, the player's decision at time τ_k is conditioned on the cards observed up to time $\tau_k - 1$. Furthermore, the player action selection is probabilistic and derived from a softmax distribution over logits. Hence, for ℓ_a being the logit score for any valid action $a \in \mathcal{A}_{\tau_k}$, the probability of selecting that action is:

$$\sigma_L(a \mid \boldsymbol{c}_{k,\tau_k-1}^{\text{partial}}, \boldsymbol{\theta}^{\text{Decision}}) = \frac{\exp(\ell_a)}{\sum_{a' \in \mathcal{A}_{\tau_t}} \exp(\ell_{a'})}.$$

The selected action a^* corresponds to the action with the highest probability, that is, $a^* = \text{argmax}_{a \in \mathcal{A}_{\tau_k}} \sigma_L\left(a \mid \boldsymbol{c}_{k,\tau_k-1}^{\text{partial}}, \boldsymbol{\theta}^{\text{Decision}}\right)$. Invalid actions (i.e., $a \notin \mathcal{A}_{\tau_t}$) are assigned $\ell_a = -\infty$, ensuring a zero probability is attributed to the invalid action. Now the interface between a model and the player action is undergone through this control vector for which $\mathbf{ct} : (\boldsymbol{c}_{k,\tau_k-1}^{\text{partial}}, \boldsymbol{\theta}^{\text{Decision}}) \to \mathbf{model}\left(\boldsymbol{\Omega}\left(\boldsymbol{c}_{k,\tau_k-1}^{\text{partial}}\right), \boldsymbol{\theta}^{\text{Decision}}\right) \xrightarrow{\sigma_L(.)} a^* \in \mathcal{A}_{\tau_t}$ where $\boldsymbol{\theta}^{\text{Decision}}$ is fixed throughout all $k = 1, 2, \ldots, K$ blackjack hands for all time

$\tau_k \leq T_k^{(p)}$, and all player decisions/actions are based on this fixed parameterization $\boldsymbol{\theta}^{\text{Decision}}$. In the framework of using a neural network as our model, we define $\boldsymbol{\Omega} : \boldsymbol{c}_{k,\tau_k-1}^{\text{partial}} \to \mathbf{a}^0 \in \mathbb{R}^{d_0}$ which signifies the vector of input nodes for time $\tau_k \leq T_k^{(p)}$. Furthermore, $\boldsymbol{\theta}^{\text{Decision}}$ are the weights and biases of the neural network, $\mathbf{w}^{\text{Decision}} \in \mathbb{R}^R$.

1.2.1 Feature engineering

We construct the feature vector using three inputs: the player's hand total at time $\tau_k - 1$, the dealer's visible upcard for the k^{th} hand, and a binary indicator denoting the presence of a usable ace in the player's hand - defined as an ace valued at 11 rather than 1. This representation aligns with the standard structure of established blackjack strategy tables, which prescribe optimal actions based on this triplet of information. By adopting this input configuration, we leverage a format that has been extensively validated through decades of empirical and theoretical research in blackjack literature.

Hence, our 1^{st} input node is defined as $a_1^0 = \frac{h\left(\mathbf{c}_{k,1:(\tau_k-1)}^{(p)}\right)}{21}$ which represents the player's hand-value at time $\tau_t - 1$, before the player makes a decision at time τ_k, normalized by 21. The 2^{nd} input node is given by $a_2^0 = \frac{h\left(c_k^{(d,\text{up})}\right)}{10}$ which encodes the dealer's upcard value as a fraction of 10. Our 3^{rd} input node is a binary indicator: $a_3^0 = \mathbb{I}\left(1 \in \mathbf{c}_{k,1:(\tau_k-1)}^{(p)} \ \cap \ \left(\sum_{c \in \mathbf{c}_{k,1:(\tau_t-1)}^{(p)}} c + 10\right) \leq 21\right)$ which denotes the presence of a usable ace in the player's hand up to time $\tau_k - 1$.

1.3 The arbitrary objective

Consider an arbitrary objective where, for a given parameter configuration $\boldsymbol{\theta} = \boldsymbol{\theta}^{\text{Decision}} \in \mathbb{R}^R$ and after playing K number of blackjack hands, we record the ROI, where the reward for the k^{th} blackjack hand is denoted as $s_k(\boldsymbol{\theta}) \cdot bet_k$ ($bet_k = 1$ is the initial bet-size for each k^{th} blackjack hand for simplicity, and scales accordingly for split and double-down actions). Hence:

$$\begin{aligned}\underset{\boldsymbol{\theta}}{\text{argmax}}\, \text{Obj}(\boldsymbol{\theta}) &= \underset{\boldsymbol{\theta}}{\text{argmax}}\, \frac{\sum_{k=1}^{K} \left(s_k(\boldsymbol{\theta}) \cdot bet_k\right)}{\sum_{k=1}^{K} bet_k} \\ &= \underset{\boldsymbol{\theta}}{\text{argmax}}\, \text{ROI}(\boldsymbol{\theta}).\end{aligned}$$

By including L2 regularization, our L2 penalized objective becomes:

$$\underset{\boldsymbol{\theta}}{\text{argmax}} \left(\text{ROI}(\boldsymbol{\theta}) - \nu \|\boldsymbol{\theta}\|^2\right). \tag{1}$$

1.4 Effects of regularization

To train the blackjack agent, we simulate a sequence of $K = 1000$ hands of blackjack - termed a "night". The training set is fully determined by an initial seed ω_0^{Train} that governs the shuffle of the D_0-deck shoe at the start of the training night. Formally, we define a sequence of the D_0 deck shoe as $\mathcal{S}(\omega)$ - indicating a specific permutation of the D_0-deck shoe, where ω is a random seed. The initial training shoe is then $\mathcal{S}(\omega_0^{\text{Train}})$. Our shoe utilizes a reshuffle threshold (deck penetration) of 50%, hence the shoe is reshuffled whenever the number of remaining cards falls below $\frac{1}{2}52 \cdot D_0$. Each reshuffle occurs by advancing the random seed deterministically using a known rule (for example, $\omega_{i+1} = f(\omega_i)$, for some deterministic function f), hence ensuring reproducibility across runs. Thus, while each reshuffle produces a distinct card ordering, the sequence of reshuffles is deterministic and reproducible given the initial seed ω_0.

Now, we define our test set as $10,000$ nights where each night is initialized with a distinct shoe shuffle $\mathcal{S}(\omega_{0,n}^{\text{Test}})$ for night n where $\omega_{0,n}^{\text{Test}} \neq \omega_0^{\text{Train}}$ for all $n = \{1, \ldots, 10000\}$. The blackjack rules still enforce a deck penetration of 50%, with reshuffling triggered accordingly, hence we define ω_i^{Train} and $\omega_{i,n}^{\text{Test}}$ to be the seed values used for reshuffling in the training night and test nights respectively. Hence to ensure no shoe permutations in the test nights overlap with or is derived from those in training night, we ensure $\omega_i^{\text{Train}} \neq \omega_{i,n}^{\text{Test}}$ implying $\mathcal{S}(\omega_i^{\text{Train}}) \neq \mathcal{S}(\omega_{i,n}^{\text{Test}})$ $\forall i, n$, ensuring disjoint training and test environments.

Furthermore, to evaluate the impact of the regularization strength ν, we apply the estimator $\hat{\boldsymbol{\theta}}_{\nu}^{\text{GA}}$ to the test set and assess both in-sample and out-of-sample hit-rates, as reported in Table 1, across various values of ν. Additionally, the mean $\mu_{ROI\%}$ and standard deviation $\sigma_{ROI\%}$ of ROI distributions (normally distributed as illustrated in Figure 2) are reported for the $10,000$ out-of-sample nights. The corresponding player decision tables are presented in Figure 1, from which we observe that the regularization strengths yielding the highest hit-rates, on the test set, tend to produce decision policies that closely resemble the S17 strategy, or more broadly, strategies anchored around a hand-value threshold of 17. Nevertheless, none of the tested regularization strengths result in a hit-rate that surpasses those achieved by the standard S17, H17, or Basic Strategy policies reported in Table 2. Furthermore, no consistent or interpretable pattern emerges in the player decision tables as the regularization strength ν is varied; the resulting policies appear to change irregularly across different values of ν. Notably, Table 1 reveals a positive association between higher hit rates and improved $\sigma_{ROI\%}$ (despite all policies exhibiting negative $\sigma_{ROI\%}$). A notably striking observation is that the standard deviations of the ROI distributions, $\sigma_{ROI\%}$, for all solutions in Table 1 are approximately the same, around 3.1%, except for $\hat{\boldsymbol{\theta}}_{\nu=0.0009}^{\text{GA}}$, which exhibits a slightly higher $\sigma_{ROI\%}$ and, coincidentally, the lowest $\mu_{ROI\%}$. The decision tables suggests this may be due to it being the only solution that recommended doubling-down on certain hands.

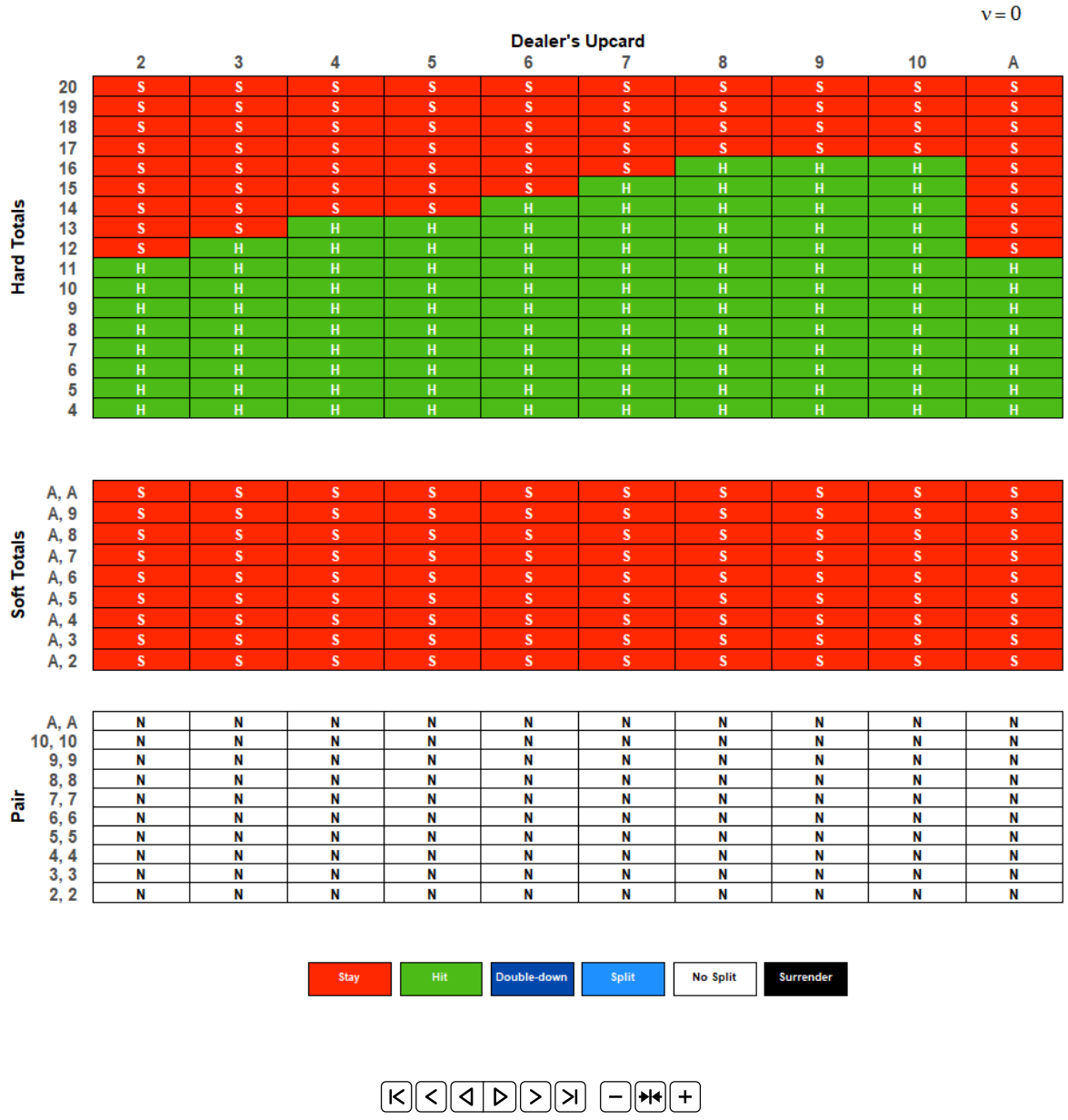


Figure 1: Player decision tables for varying ν (removal of surrender table seeing as, given the range of ν, player never opts to surrender).

	In-Sample		Out-of-Sample		
Regularization Strength ν	Hit-Rate	ROI %	Hit-Rate	$\mu_{ROI\%}$	$\sigma_{ROI\%}$
0.0000	0.4868	−0.1500	**0.4575**	**-5.5064**	3.1067
0.0001	0.4708	−2.8000	0.4245	−11.9999	3.1051
0.0002	0.4840	−0.6500	0.4464	−7.7526	3.1170
0.0003	0.4397	−9.1000	0.4222	−12.3205	3.0765
0.0004	0.4695	−3.1500	**0.4535**	**-6.2797**	3.1213
0.0005	0.4542	−5.9500	0.4215	−12.5486	3.1114
0.0006	0.4274	−11.7500	0.4058	−15.8433	3.1615
0.0007	0.4490	−7.5500	0.4342	−9.9631	3.1131
0.0008	0.4836	−0.6500	**0.4543**	**-6.0970**	3.1112
0.0009	0.4343	−8.6500	0.4098	−16.5464	3.5682

Table 1: Regularization strength ν vs. hit-rate (in- and out-of-sample).

	In-Sample		Out-of-Sample		
Decision Policy	Hit-Rate	ROI %	Hit-Rate	$\mu_{ROI\%}$	$\sigma_{ROI\%}$
Purely Random	0.2704	-37.85	0.2597	-41.8837	3.1961
Random Stay/Hit	0.3442	-27.35	0.3331	-29.6443	3.0478
S17	0.4266	-11.05	0.4578	-5.3242	3.0711
H17	0.4190	-12.35	0.4596	-5.0095	3.0877
Basic Strategy	0.4430	-4.25	0.4630	-0.6716	3.5657

Table 2: Hit-rate and ROI performance of common blackjack decision policies.

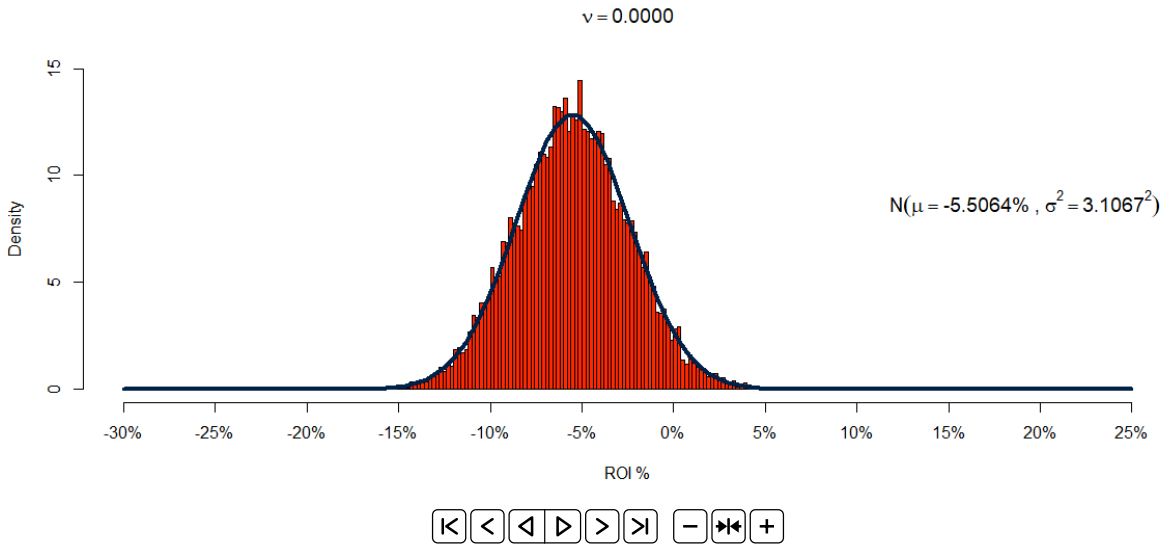


Figure 2: Out-of-sample ROI distributions for varying ν, corresponding to Table 1 and ROI distributions of common blackjack decision policies corresponding to Table 2.

2 The Blackjack Problem II: Controlling Bet Size

In this section, we focus exclusively on controlling the bet size (placed by the player before any cards are drawn), without influencing the player's in-game decisions as in Section 1. Accordingly, player decisions are assumed to follow Basic Strategy, meaning the bet multiplier s_k is determined solely by Basic Strategy recommendations, rather than being governed by $\boldsymbol{\theta}^{\text{Decision}}$ as in Section 1.

2.1 Control: Bet size

We control the amount to bet for the k^{th} blackjack hand, (which occurs before any cards are dealt for the k^{th} hand) through the means of the control vector $\mathbf{ct}\left(\boldsymbol{\mathcal{H}}_{k-1}, \boldsymbol{\theta}^{\text{Bet}}\right) \in [0,1]$ for some parameter configuration $\boldsymbol{\theta}^{\text{Bet}} \in \mathbb{R}^S$ (note the index for the card history vector $\boldsymbol{\mathcal{H}}$ is $k-1$ to indicate that betting occurs before any cards are observed for the k^{th} blackjack hand). Now the interface between a model and the bet sizing is undergone through this control vector for which $\mathbf{ct} : (\boldsymbol{\mathcal{H}}_{k-1}, \boldsymbol{\theta}^{\text{Bet}}) \to \mathbf{model}\left(\boldsymbol{\Omega}\left(\boldsymbol{\mathcal{H}}_{k-1}\right), \boldsymbol{\theta}^{\text{Bet}}\right) \xrightarrow{\sigma_L(.)} [0,1]$ where $\boldsymbol{\theta}^{\text{Bet}}$ is fixed throughout all $k = 1, 2, \ldots, K$ blackjack hands, and betting decisions at each k^{th} hand are based on this fixed parameterization $\boldsymbol{\theta}^{\text{Bet}}$. In the framework of using a neural network as our model, we define $\boldsymbol{\Omega} : \boldsymbol{\mathcal{H}}_{k-1} \to \mathbf{a}^0 \in \mathbb{R}^{d_0}$ which signifies the vector of input nodes for the t^{th} hand. Furthermore, $\boldsymbol{\theta}^{\text{Bet}}$ are the weights and biases of the neural network, $\mathbf{w}^{\text{Bet}} \in \mathbb{R}^S$, and $\sigma_L(.)$ denotes a sigmoid activation function.

2.1.1 Feature engineering

To encode the current state of the game in a manner conducive to learning, we propose a feature vector that captures both (i) the true count of the remaining deck as formally defined in Appendix B and (ii) a weighted summary of the distribution of unseen cards. The true count serves as a classical card-counting statistic reflecting the favourability of the shoe, while the weighted distribution provides fine-grained information about the residual card composition.

Specifically, we define the first input node of the neural network as the scaled true count $a_1^0 = \frac{TC_{k-1}}{3}$, where TC_{k-1} is the true count at time $k-1$ normalized by a constant (here, 3) to map the feature into a compact numerical range. To complement this, we construct additional features that reflect the remaining proportion of each card value in the shoe, weighted by their nominal face value. For cards $i = 2, \ldots 9$ and $i = 11$ we define $a_i^0 = \frac{4iD_0 - i\sum_{c \in \boldsymbol{\mathcal{H}}_{k-1}} \mathbb{I}(c=i)}{4iD_0}$. This expression measures the remaining total "value-weighted mass" of card i, relative to its initial value-weighted mass across the full shoe. For ten-valued cards (10, J, Q, K), which occur with higher frequency (16 per deck), we define $a_{10}^0 = \frac{16iD_0 - 10\sum_{c \in \boldsymbol{\mathcal{H}}_{k-1}} \mathbb{I}(c=10)}{16iD_0}$. Furthermore, we define two distinct models that differ in the number of input nodes utilized. Now $\underset{(\text{I})}{\mathbf{model}}\left(a_1^0, \boldsymbol{\theta}^{(\text{I})}\right)$ relies solely on the true count to create the sole input

node a_1^0. In contrast, $\overset{(II)}{\mathbf{model}}\left(\mathbf{a}^0, \boldsymbol{\theta}^{(II)}\right)$ incorporates all 11 previously specified input nodes, which offers a richer characterization of card composition than the true count alone.

Hence, in the proposed neural network architecture, the input layer comprises of $d_0 = 11$ nodes, which collectively processes $\boldsymbol{\mathcal{H}}_{k-1}$ to produce a singular output, denoted as the betting propensity, $bet_k \in [0, 1]$ representing the relative confidence in wagering, where the actual bet size is $\tilde{bet}_k = 1 + 9 \times bet_k \in [1, 10]$, ensuring that a positive, nonzero amount is wagered on each hand (where 1 and 10 may be viewed as the table minimum and maximum bets, respectively). Additionally, in scenarios where the true count satisfies $TC_{k-1} = 0$ and the card history is empty, $|\boldsymbol{\mathcal{H}}_{k-1}| = 0$, indicating a reshuffled shoe, the betting strategy defaults to $bet_k = 0$, hence $\tilde{bet}_k = 1$. In such cases, the historical card information $\boldsymbol{\mathcal{H}}_{k-1}$ is excluded from the control vector $\mathbf{ct}$, thereby bypassing its integration into the bet-size decision process.

2.2 The arbitrary objective

Consider an arbitrary objective where, for a given parameter configuration $\boldsymbol{\theta} = \boldsymbol{\theta}^{\text{Bet}} \in \mathbb{R}^S$ and after playing T number of blackjack hands, we record the ROI, where the reward for the k^{th} blackjack hand is denoted as $s_k \cdot \tilde{bet}_k(\boldsymbol{\theta})$ (where $\tilde{bet}_k(\boldsymbol{\theta})$ scales accordingly to split and double-down actions). Hence:

$$\begin{aligned}\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \text{Obj}(\boldsymbol{\theta}) &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \frac{\sum_{k=1}^{K}\left(s_k \cdot \tilde{bet}_k(\boldsymbol{\theta})\right)}{\sum_{k=1}^{K} \tilde{bet}_t(\boldsymbol{\theta})} \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \text{ROI}\left(\boldsymbol{\theta}\right).\end{aligned}$$

By including L2 regularization, our L2 penalized objective becomes:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left(\text{ROI}\left(\boldsymbol{\theta}\right) - \nu\|\boldsymbol{\theta}\|^2\right). \tag{2}$$

2.3 Effects of regularization

To evaluate the effect of varying regularization strengths ν on the learned betting behavior bet_k, we conduct a response curve analysis. For visualization purposes, we employ $\overset{(I)}{\mathbf{model}}$ that uses a single input node a_1^0, representing the scaled true count of the shoe denoted in the previous section. The resulting response function is given by $\overset{(I)}{\mathbf{model}}\left(a_1^0, \hat{\boldsymbol{\theta}}_\nu^{\text{GA},(I)}\right) \rightarrow bet_k \in [0, 1]$ and is depicted in Figure 3.

As established in the blackjack literature (see also Appendix B), exploiting the game advantageously requires betting in proportion to the true count. The figure reveals that the solutions $\hat{\boldsymbol{\theta}}_\nu^{\text{GA},(I)}$, across most ν values considered, tend to produce elevated betting propensities bet_k predominantly at highly negative true counts. While a few values of ν do give rise to solutions which elicit high betting behavior at large positive true counts - as theoretically desirable - this behavior is not consistently observed for all solutions.

Now this undesirable behaviour may be revealed by examining Table 4, where the notion of betting proportional to the true count of the shoe is illustrated. Notably, an inconsistency arises: in-sample ROI performance of these theoretical-based strategies is surprisingly poor. This observation suggests that the training set may not be conducive to the effective training of $\hat{\boldsymbol{\theta}}_\nu^{\text{GA},(I)}$ - that is, this training set may not provide a sufficiently informative environment such that $\hat{\boldsymbol{\theta}}_\nu^{\text{GA},(I)}$ learns to bet in accordance to the true count. The theoretical betting strategy's superior performance is substantiated, however, when evaluating out-of-sample data, as presented in Table 4. Specifically, the mean ROI ($\mu_{ROI\%}$) values for the theory-driven strategies demonstrate a marked improvement, approaching break-even levels with greater consistency compared to learnt betting in Table 3.

Table 3 reveals that both $\overset{(I)}{\mathbf{model}}$ and $\overset{(II)}{\mathbf{model}}$ exhibit substantial underperformance on the out-of-sample set. On a few occasions, the solutions $\hat{\boldsymbol{\theta}}_\nu^{\text{GA}}$ yield mean ROI values ($\mu_{ROI\%}$) that are only marginally higher than those obtained from the purely random betting policy presented in Table 4. Unsurprisingly, none of the betting configurations derived from $\hat{\boldsymbol{\theta}}_\nu^{\text{GA}}$ solutions outperform any of the true count-based betting strategies on the out-of-sample set, as reported in Table 4. These findings are consistent with the earlier observation that the training set may be ill-suited for facilitating effective learning based on established blackjack theory. Furthermore, we refrain from making general claims regarding comparative performance across varying values of ν between the two models, as the two models differ in complexity with respect to their number of input nodes - $\overset{(I)}{\mathbf{model}}$ utilizing a_1^0 as its feature vector and $\overset{(II)}{\mathbf{model}}$ utilizing $\mathbf{a}^0$ as its feature vector.

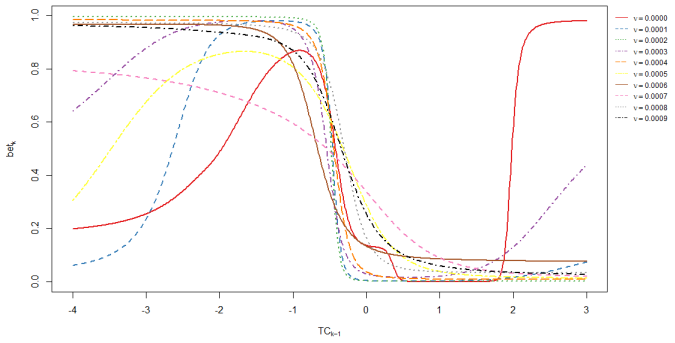


Figure 3: bet_k vs TC_{k-1} for various ν using $\overset{(I)}{\textbf{model}}\left(a_1^0, \hat{\boldsymbol{\theta}}_\nu^{\mathrm{GA},(I)}\right)$

.

	$\overset{(I)}{\textbf{model}}\left(a_1^0, \hat{\boldsymbol{\theta}}_\nu^{\mathrm{GA},(I)}\right)$			$\overset{(II)}{\textbf{model}}\left(\mathbf{a}^0, \hat{\boldsymbol{\theta}}_\nu^{\mathrm{GA},(II)}\right)$		
	In-Sample	Out-of-Sample		In-Sample	Out-of-Sample	
Regularization Strength ν	ROI %	$\mu_{ROI\%}$	$\sigma_{ROI\%}$	ROI %	$\mu_{ROI\%}$	$\sigma_{ROI\%}$
0.0000	0.7440	**-0.7650**	4.4523	14.6180	−0.9887	5.5638
0.0001	0.7670	−1.1536	5.1288	11.4806	−0.9207	5.2493
0.0002	−0.5578	−1.3047	5.1894	8.1637	−0.8404	4.7393
0.0003	−0.3340	−1.1145	4.7978	6.0955	−0.8418	5.0195
0.0004	−0.9778	−1.2743	4.9373	−2.0912	−0.8334	3.9489
0.0005	−0.9952	−1.1020	4.3136	−0.3641	−1.1610	4.4662
0.0006	−1.9658	−1.1819	4.5306	−3.9218	−0.6722	3.6046
0.0007	−2.6602	−1.0656	4.0409	−2.2319	**-0.4890**	4.1832
0.0008	−1.4639	−1.1926	4.5497	8.74736	−0.7584	4.8677
0.0009	−1.8479	−1.1604	4.3676	−3.3625	−0.7375	3.6303

Table 3: ROI in-sample and out-of-sample performance across regularization strengths ν using $\overset{(I)}{\textbf{model}}\left(a_1^0, \hat{\boldsymbol{\theta}}_\nu^{\mathrm{GA},(I)}\right)$ and $\overset{(II)}{\textbf{model}}\left(\mathbf{a}^0, \hat{\boldsymbol{\theta}}_\nu^{\mathrm{GA},(II)}\right)$.

	In-Sample	Out-of-Sample	
Betting Policy	ROI %	$\mu_{ROI\%}$	$\sigma_{ROI\%}$
Purely Random	−0.2202	−0.6909	4.0185
$TC_{k-1} > 0$	−5.8488	−0.0012	5.0031
$TC_{k-1} > 1$	−4.4912	0.0511	5.5327
$TC_{k-1} > 2$	−1.8258	−0.0233	6.0224
$TC_{k-1} > 3$	−5.9491	−0.2950	5.7936

Table 4: ROI in-sample and out-of-sample performance of betting policies whose bet sizes are linearly proportional to TC_{k-1} ($bet_k = \frac{TC_{k-1}}{3} \cdot \mathbb{I}(TC_{k-1} > x)$ for $x = 0, 1, 2, 3$).

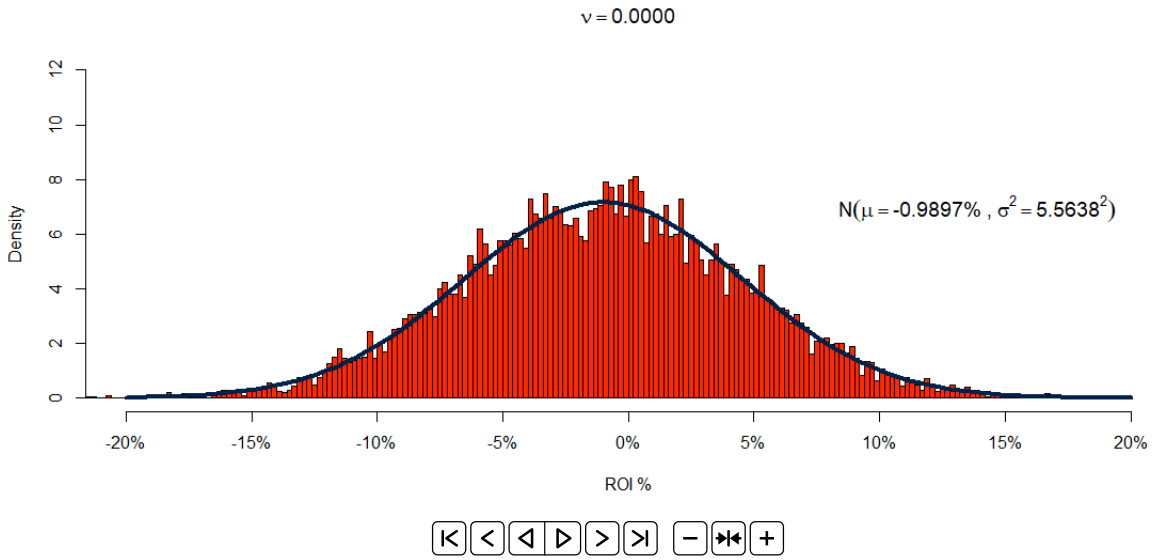
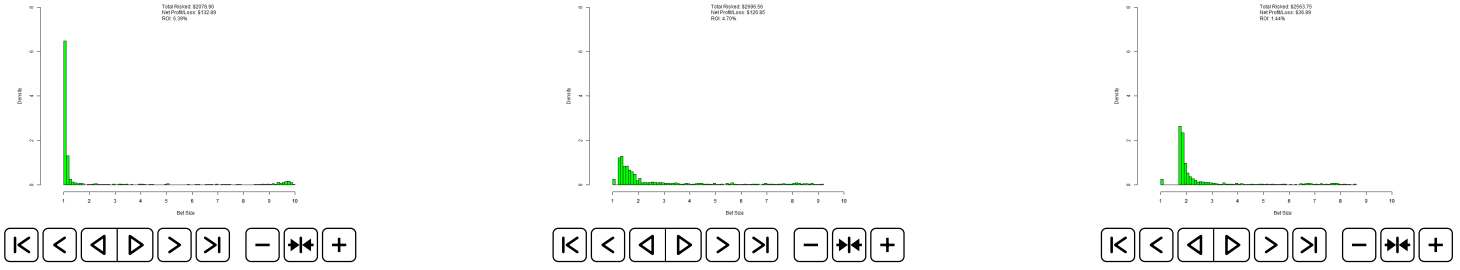


Figure 4: Out-of-sample ROI distributions for varying ν using $\overset{(\mathrm{II})}{\mathbf{model}}\left(\mathbf{a}^0, \hat{\boldsymbol{\theta}}_{\nu}^{\mathrm{GA},(\mathrm{II})}\right)$, corresponding to Table 3 and ROI distributions of common blackjack betting policies, corresponding to Table 4.

Now, Figure 5 illustrates the distributions of bet sizes, $\tilde{bet}_k \in [1, 10]$, throughout a night using $\overset{(\mathrm{II})}{\mathbf{model}}\left(\mathbf{a}^0, \hat{\boldsymbol{\theta}}_{\nu}^{\mathrm{GA},(\mathrm{II})}\right)$, for 100 nights, where the net profit or loss is computed as the cumulative gain or loss between the 1,000 hands of blackjack played at the end of each night. The optimisation process consistently yields solutions that exhibit sparse betting patterns - that is, the player seems to be consistently opted to bet close to the table minimum, for these specific ν.



$\nu = 0$ $\nu = 0.0005$ $\nu = 0.0007$

Figure 5: Distributions of bet size $\tilde{bet}_k$ over 100 nights for various ν.

3 Hybrid

Within the two-sample MCMC framework described in Section **??**, the incorporation of pseudo-likelihoods introduces a sharpness parameter $\beta \in \mathbb{R}^+$ that governs the peakedness of the pseudo-likelihood. Consequently, this parameter also influences the shape of the conditional posterior $p(\boldsymbol{\theta} \mid \mathcal{D}, \sigma_\theta^2)$, as increased values of β lead the pseudo-likelihood to dominate the conditional, resulting in a more sharply peaked distribution - an effect discussed in greater detail in Section **??**. An increase in β renders the conditional posterior $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$ more sensitive to variations in the objective function, thereby producing a sharply peaked distribution concentrated around a dominant mode. In other words, the heightened peakedness intensifies the concentration of samples in the vicinity of this dominant mode.

Now, to leverage this property, it is proposed to replace the MH sampling step in Section **??**, which samples $\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D}$, with an iterative optimisation method. Given the sharply peaked nature of the conditional posterior, induced by a large β, the latter stages of an optimisation method converge to the mode of $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$ - given the optimisation method explicitly seeks $\operatorname{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$. These iterates can be regarded as samples from the approximate modal region of the conditional posterior, particularly when the posterior is highly concentrated around a dominant mode.

Additionally, Section **??** and **??** highlighted that variations in likelihood sharpness, the initial dispersion σ_{Init}^2, and the particular pseudo-likelihood formulation employed during MCMC, substantially influenced the dispersion parameter σ_θ^2, and by extension, the degree of regularization inferred - given that $\sigma_\theta^2 \propto \frac{1}{\nu}$. This raised a critical concern: does the hierarchical Bayesian structure (implemented via two-sample MCMC) truly enable data-driven regularization, or does it instead reintroduce user-specified regularization? After all, the user must still choose the pseudo-likelihood formulation (and its associated sharpness parameter β), as well as the initial dispersion σ_{Init}^2. In this light, it becomes necessary to reconsider the idea that the training data determines a meaningful level of regularization embedded within the MAP estimates.

Considering the iterative optimisation is to be employed rather than MH sampling in Block 1 of the two-sample MCMC framework, we are inclined to reinterpret the sampling of $\sigma_\theta^2 \mid \boldsymbol{\theta}, \mathcal{D}$ in Block 2 as an auxiliary mechanism for enhancing exploration within the optimisation process. Specifically, since the objective function being maximised is the conditional posterior $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$, the act of sampling $\sigma_\theta^2 \mid \boldsymbol{\theta}, \mathcal{D}$ at each iteration induces a fluctuating optimisation landscape. This variability results in a "wobbly" optimisation trajectory, wherein the shape of the objective function changes slightly from one iteration to the next. Such non-static behavior naturally encourages broader exploration of the parameter space before the algorithm converges toward a dominant mode of the conditional distribution. Furthermore, it must be noted that the sampling of $\sigma_\theta^2 \mid \boldsymbol{\theta}, \mathcal{D}$ per iteration is still giving the training data control over what level of regularisation to infer over the MAP estimates, as before.

3.1 Genetic algorithm hybrid

Under the framework of a GA, where our fitness scores reflect that of maximising the conditional $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$, we can view each generation as a pursuit to refine $\boldsymbol{\theta} \in \mathbb{R}^S$ such that it closely reflects $\text{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta} \mid \mathcal{D}, \sigma_\theta^2)$, that is, the mode of $p(\boldsymbol{\theta} \mid \mathcal{D}, \sigma_\theta^2)$. Furthermore, we may propose the notion that a GA can be viewed as a means of drawing $\boldsymbol{\theta}$ from an approximate modal region of $p(\boldsymbol{\theta} \mid \mathcal{D}, \sigma_\theta^2)$, where each n^{th} individual per m^{th} generation, denoted as $\boldsymbol{\theta}^{(n,m)}$, may be viewed as a draw.

Now the fitness values for $\boldsymbol{\theta}^{(n,m)}$:

$$\begin{aligned}
f_{n,m} &= \max\left\{p\left(\boldsymbol{\theta}^{(n,m-1)} \mid \sigma^2_{\theta^{(n,m-1)}}, \mathcal{D}\right)\right\} \\
&= \max\left\{p\left(\mathcal{D} \mid \boldsymbol{\theta}^{(n,m-1)}\right) p\left(\boldsymbol{\theta}^{(n,m-1)} \mid \sigma^2_{\theta^{(n,m-1)}}\right)\right\} \\
&= \max\left\{p\left(\mathcal{D} \mid \boldsymbol{\theta}^{(n,m-1)}\right) \cdot \frac{1}{\sqrt{\left(2\pi\sigma^2_{\theta^{(n,m-1)}}\right)^S}} \exp\left(-\frac{1}{2\sigma^2_{\theta^{(n,m-1)}}}\|\boldsymbol{\theta}^{(n,m-1)}\|^2\right)\right\} \\
&= \max\left\{\log\left(p\left(\mathcal{D} \mid \boldsymbol{\theta}^{(n,m-1)}\right)\right) - \frac{1}{2\sigma^2_{\theta^{(n,m-1)}}}\|\boldsymbol{\theta}^{(n,m-1)}\|^2 - \frac{S}{2}\log\left(2\pi\sigma^2_{\theta^{(n,m-1)}}\right)\right\}.
\end{aligned}$$

Afterwhich we conduct the sampling step in Block 2 in Section **??**, where $\left(\sigma_\theta^2\right)^{(n,m)} \mid \boldsymbol{\theta}^{(n,m)}, \mathcal{D} \sim \text{Inv-Gamma}\left(a + \frac{S}{2}, b + \frac{\|\boldsymbol{\theta}^{(n,m)}\|^2}{2}\right)$.

3.2 Gradient descent hybrid for classification

Furthermore, for non-arbitrary objectives, that is for problems whose cost functions can be directly related to input-ouput pairs (for example, in supervised learning), which give rise to data-driven likelihoods (and not pseudo-likelihoods as we've been employing in this study), we may employ gradient-based optimisation methods in order to maximise the conditional $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$. We propose that we may adopt the same approach as before, by postulating that each iteration of gradient descent may be viewed as sampling from the approximate modal region of the conditional $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$. Hence, the m^{th} iteration of $\boldsymbol{\theta}$ for $m = 1, 2 \ldots, M$ with step-size h is updated as:

$$\boldsymbol{\theta}^{(m)} = \boldsymbol{\theta}^{(m-1)} - h \cdot \nabla \text{Obj}\left(\boldsymbol{\theta}^{(m-1)}, \left(\sigma_\theta^2\right)^{(m-1)}\right). \tag{3}$$

As gradient descent seeks to minimise an objective function (Obj), we may express our optimisation problem, $\text{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$, as:

$$\underset{\boldsymbol{\theta}}{\text{argmin}}\, \text{Obj}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\text{argmin}} \left\{-\log\left[p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})\right]\right\}.$$

Now,

$$\begin{aligned}
-\log\left[p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})\right] &\propto -\log\left[p\left(\mathcal{D} \mid \boldsymbol{\theta}\right)\right] - \log\left[p\left(\boldsymbol{\theta} \mid \sigma_\theta^2\right)\right] \\
&\propto -\log\left[p\left(\mathcal{D} \mid \boldsymbol{\theta}\right)\right] + \frac{1}{2\sigma_\theta^2}\|\boldsymbol{\theta}\|^2 + \frac{S}{2}\log\left(2\pi\sigma_\theta^2\right).
\end{aligned} \tag{4}$$

In the context of K-class classification, we know a neural network outputs logits $z_{i,1}, z_{i,2}, \ldots z_{i,d_L}$ for observations $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, d_L = K$ classes. These are transformed into probabilities using a softmax activation function, hence, for a given observation i, the predicted probability of class k is:

$$p_{i,k} = \sigma_L(z_{i,k}) = a_k(i)^L = \frac{\exp(z_{i,k})}{\sum_{j=1}^{d_L}\exp(z_{i,j})}.$$

So for a one-hot-encoded y_i, $p(y_i) = \prod_{k=1}^{d_L}\left(a_k(i)^L\right)^{y_{i,k}}$. Now for N independent observations, the likelihood is:

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{k=1}^{d_L}\left(a_k(i)^L\right)^{y_{i,k}}. \tag{5}$$

Being such, we substitute this likelihood into Equation 4 to obtain our cost function:

$$\begin{aligned}
\mathcal{L}\left(\boldsymbol{\theta}, \sigma_\theta^2\right) = -\log\left[p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})\right] &\propto \underbrace{\left(-\sum_{i=1}^{N}\sum_{k=1}^{d_L} y_{i,k}\log\left(a_k(i)^L\right) + \frac{1}{2\sigma_\theta^2}\|\boldsymbol{\theta}\|^2\right)}_{\mathcal{L}_{\text{cross-entropy}}(\boldsymbol{\theta}) + \text{L2 penalty}} + \frac{S}{2}\log\left(2\pi\sigma_\theta^2\right) \\
&= \mathcal{L}^*_{\text{cross-entropy}}(\boldsymbol{\theta}, \sigma_\theta^2) + \frac{S}{2}\log\left(2\pi\sigma_\theta^2\right).
\end{aligned} \tag{6}$$

Gradient descent proceeds via backpropagation, with the gradient of our cost function satisfying $\nabla\mathcal{L}(\boldsymbol{\theta}, \sigma_\theta^2) = \nabla\mathcal{L}^*_{\text{cross-entropy}}(\boldsymbol{\theta}, \sigma_\theta^2)$ since the additional term in Equation 6 does not depend on $\boldsymbol{\theta}$ seeing as $\nabla = \frac{\partial(\cdot)}{\partial\boldsymbol{\theta}}$. Now after sampling $\boldsymbol{\theta}^{(m)}$ from Equation 3, we sample our dispersion parameter as $(\sigma_\theta^2)^{(m)} \mid \boldsymbol{\theta}^{(m)}, \mathcal{D} \sim \text{Inv-Gamma}\left(a + \frac{S}{2},\ b + \frac{\|\boldsymbol{\theta}^{(m)}\|^2}{2}\right)$ as before. Now under the framework of GD, for a fixed σ_θ^2, our cost function (Equation 6) is static, and GD follows a smooth, deterministic trajectory toward a minimum. By sampling σ_θ^2 per iteration, we create a "wobbly" optimisation path: the gradient direction shifts not just due to the current $\boldsymbol{\theta}$, but also because the regularization term's influence (that is, the influence of σ_θ^2) fluctuates. With fixed σ_θ^2, GD descends a static trough - a single, well-defined valley - to its lowest point. Varying σ_θ^2 reshapes this trough each iteration: large σ_θ^2 (low ν) widens and shallows it, letting the algorithm wander; small σ_θ^2 (high ν) narrows and deepens it near the origin, tugging inward. The descent becomes a pursuit of a shifting bottom, possibly broadening the exploration before homing in.

3.3 Toy example: 3-class classification

We apply the three previously described optimisation methods - namely, MCMC, GD Hybrid and GA Hybrid - to the 3-class classification problem introduced in Appendix C, using the likelihood formulation given in Equation 5. To ensure comparability across methods, we initialize all algorithms with the same starting solution, denoted as $\boldsymbol{\theta}^{(1)}$. For the GA Hybrid approach, every individual in the initial population was set to this same value - that is $\boldsymbol{\theta}^{(n,1)} = \boldsymbol{\theta}^{(1)}$ for all n individuals. Each method, however, comes with specific caveats that influence its behavior. For the GA Hybrid method, several hyperparameters significantly affect the optimisation dynamics. In particular, the choice of lower and upper bounds constraining the search space directly impacts the magnitude of $\|\boldsymbol{\theta}\|^2$. Additionally, the mutation probability applied to a particular gene in a parent solution governs the extent of variability of $\|\boldsymbol{\theta}\|^2$. With respect to MCMC, the effect of the initial dispersion parameter σ^2_{Init} on performance has already been examined in Section **??**. For the GD Hybrid approach, it was found that variation in the step size h primarily influenced the convergence rate.

Figure 6 illustrates that the different optimisation methods give rise to varying concentrations in the marginal distributions of the dispersion parameter σ_θ^2, which we attribute to differences in the degree of regularization implicitly induced by each method. This variation stems from the distinct convergence behaviors of $\|\boldsymbol{\theta}^{(j)}\|^2$ across iterations. Recall that $\sigma_\theta^2 \mid \boldsymbol{\theta}, \mathcal{D} \sim \text{Inv-Gamma}\left(a + \frac{S}{2},\ b + \frac{\|\boldsymbol{\theta}\|^2}{2}\right)$, with $a, b \approx 0$. Hence it follows that if $\|\boldsymbol{\theta}^{(j)}\|^2$ fluctuates around a constant value c, then the marginal distribution of $\sigma_\theta^2 \mid \mathcal{D}$ should also follow an inverse-gamma distribution with approximately constant shape and scale parameters as such: $\sigma_\theta^2 \mid \mathcal{D} \sim \text{Inv-Gamma}\left(a + \frac{S}{2},\ b + \frac{c}{2}\right)$. Notably, the GD Hybrid method exhibits the least variability in $\|\boldsymbol{\theta}^{(j)}\|^2$, followed by MCMC, both of which result in marginal distributions of $\sigma_\theta^2 \mid \mathcal{D}$ that closely follow an inverse-gamma form. Among the optimisation methods considered, the GA Hybrid approach is the only one that results in a clearly non-inverse-gamma marginal of $\sigma_\theta^2 \mid \mathcal{D}$, which we attribute to the greater variability in $\|\boldsymbol{\theta}^{(j)}\|^2$ across iterations. This variation suggests that the GA explores more diverse regions of the parameter space, which in turn leads to a more dispersed marginal of $\sigma_\theta^2 \mid \mathcal{D}$ (as noted earlier, this variability can be modulated through the mutation probability).

Figure 7 presents the distribution of one of the estimated parameters - specifically $\hat{\theta}_2$. The top panel displays the results on a common scale to highlight the extent to which the different optimisation methods produce distinct solutions, $\hat{\boldsymbol{\theta}}$, allowing for a direct comparison between them. In contrast, the bottom panel uses individual scales to better visualize the shape of each distribution of $\hat{\theta}_2$. This latter view illustrates that both hybrid methods can function as approximate sampling techniques, all yielding unimodal distributions for the estimated parameter, $\hat{\theta}_2$ (albeit with differing spreads). While MCMC serves as the baseline for comparison - since it directly samples from the conditional posterior $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$ rather than sampling from the approximate modal region of the posterior - multiple runs of the sampler often yield noticeably different parameter solutions. This behavior is also observed in the other two optimisation methods as well. Nevertheless, these disparate solutions tend to produce comparable in-sample performance, suggesting the existence of a multi-modal posterior landscape. Accordingly, one cannot conclude that the methods are fundamentally dissimilar solely on the basis of differences in the estimated solutions $\hat{\boldsymbol{\theta}}$, as variation in solutions arises even within a single optimisation method across multiple runs.

Finally, Figure 8, in conjunction with Table 5, illustrates the performance of the three optimisation methods. The response curves in Figure 8 reveal substantial similarity across methods, with each successfully delineating distinct classification regions for the three particle types. Moreover, Table 5 reports comparable in- and out-of-sample performance across the three approaches, further supporting the notion that all methods achieve a similar level of predictive accuracy.

It is important to recall that such hybrid optimisation methods are comparable to that of full MCMC, only in settings where the conditional posterior $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$ is highly peaked. The similarity in results across the three methods in this case may therefore indicate that, for the data under consideration, the conditional posterior is indeed sharply concentrated. However, this observation is likely data-specific and should not be taken as evidence that such comparability will generalize across all datasets. In the following section, we intentionally increase the sharpness of the likelihood function - and, by extension, the conditional posterior - to investigate the behaviour of the optimisation methods under a sharpened conditional.

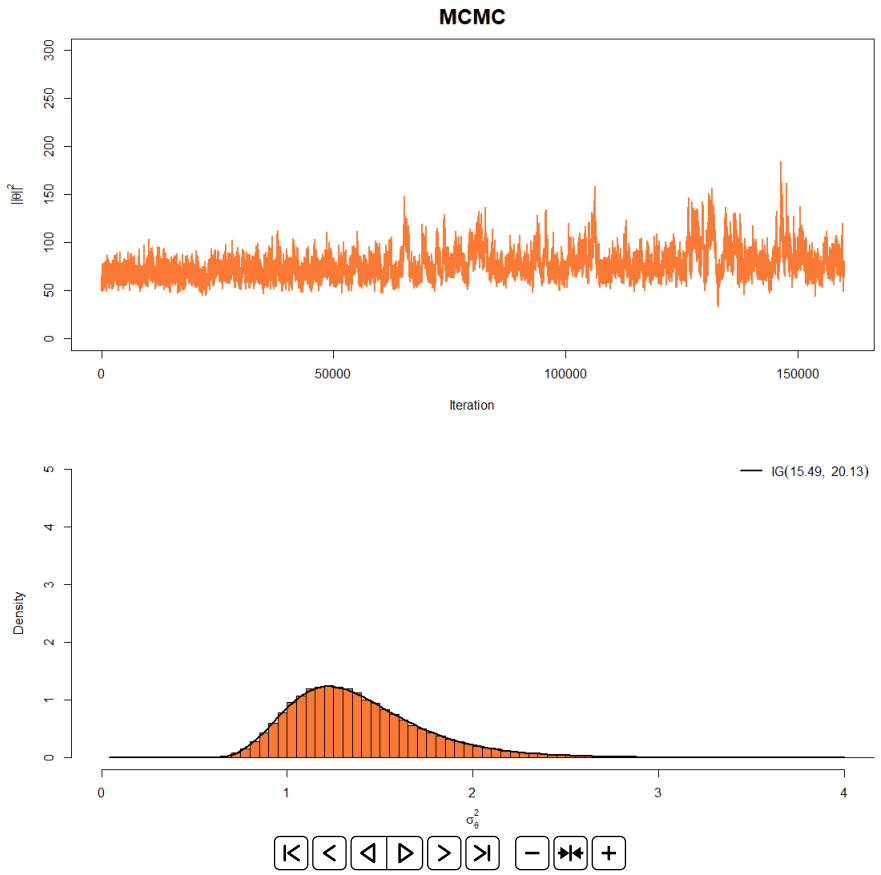


Figure 6: $\|\boldsymbol{\theta}^{(j)}\|^2$ for $j = 1, \ldots 150,000$ (post burn-in) with distribution of marginal of $\sigma^2_\theta \mid \mathcal{D}$.

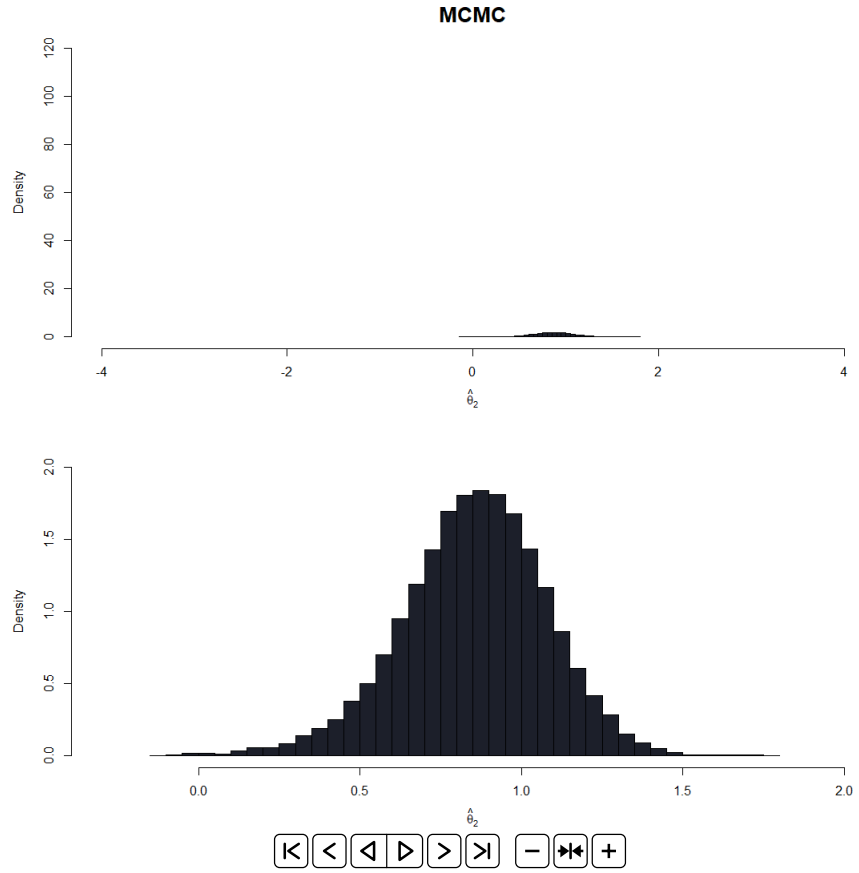


Figure 7: Distribution of $\hat{\theta}_2$. The top panel uses a common scale across all three optimisation methods to facilitate direct comparison, while the bottom panel employs different scales across methods.

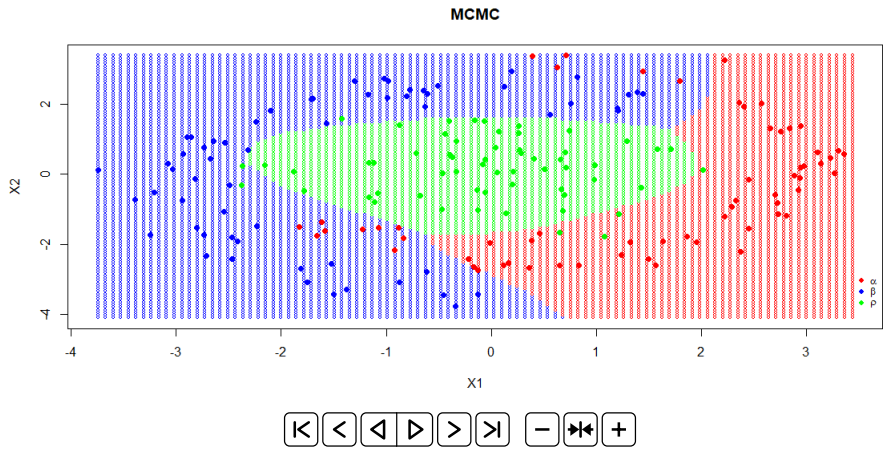


Figure 8: Response curve for different optimisation methods, represented as a cross-sectional heat map of particle classification regions (circles represent the out-of-sample particles).

Optimisation Method	In-sample	Out-of-sample
MCMC	86.67	86.67
GD Hybrid	85.55	87.22
GA Hybrid	82.78	81.11

Table 5: Accuracy for in-sample and out-of-sample sets using MCMC and two hybrid optimisation techniques.

3.4 Hybrid for Blackjack Problem I, II and III

This section primarily examines the similarities and differences between solutions derived from the two-sample MCMC framework (as outlined in Section **??**) and the Hybrid method previously introduced, when applied to Blackjack Problems I and II (using $\underset{\text{(II)}}{\textbf{model}}$ for II). In addition, we introduce a third variant, referred to as Blackjack Problem III, in which both the decision-making and bet-sizing parameter sets are embedded into a unified parameter vector: $\boldsymbol{\theta} = \left[\boldsymbol{\theta}^{\text{Bet}\prime}, \boldsymbol{\theta}^{\text{Decision}\prime}\right] \in \mathbb{R}^{S+R}$. In this formulation, $\boldsymbol{\theta}$ contains the weights and biases governing two distinct neural networks: one dedicated to decision-making and the other to bet-sizing (noting we use $\underset{\text{(II)}}{\textbf{model}}$ for the bet-sizing network). The overarching objective for Blackjack Problem III remains unchanged, namely, to maximise the ROI achieved at the conclusion of the training night, such that:

$$\begin{aligned}\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,\text{Obj}(\boldsymbol{\theta}) &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,\frac{\sum_{k=1}^{K}\left(s_k\left(\boldsymbol{\theta}\right)\cdot\tilde{bet}_k(\boldsymbol{\theta})\right)}{\sum_{k=1}^{K}\tilde{bet}_t(\boldsymbol{\theta})}\\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,\text{ROI}\left(\boldsymbol{\theta}\right).\end{aligned}$$

We employ the exponential likelihood across all three blackjack problems, albeit with differing sharpness parameters, β, selected to yield the most favourable MCMC convergence. For fairness in comparison, the corresponding Hybrid method implementations employ identical β values. Accordingly, the pseudo-likelihood is given by $p(\mathcal{D} \mid \boldsymbol{\theta}) = \exp(\beta \cdot \text{ROI}(\boldsymbol{\theta}))$. Now, given that our objective function is arbitrary, we restrict our analysis to the GA Hybrid method described in Section 3.1, and do not employ the GD Hybrid method from Section 3.2, as the objective function in question is non-differentiable with respect to $\boldsymbol{\theta}$.

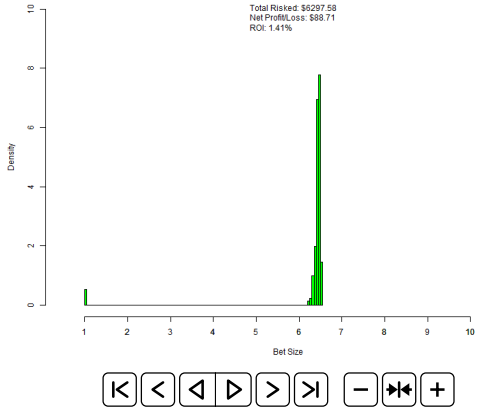
Table 6 presents the results of the two-sample MCMC optimisation method alongside its corresponding GA Hybrid method for each of the three Blackjack problems. In the case of Blackjack Problem I, both methods favour the stand action for all soft and hard totals, and never opt to surrender or split pairs - this being the reason why we obtain identical in- and out- of sample performance.

For Blackjack Problem II, we observe near-identical in-sample and out-of-sample performance. As illustrated in Figure 9, this similarity appears to stem from both methods producing agents with comparable betting behaviour - specifically, agents that concentrate their bet-size to approximately 6 to 8 units per night.

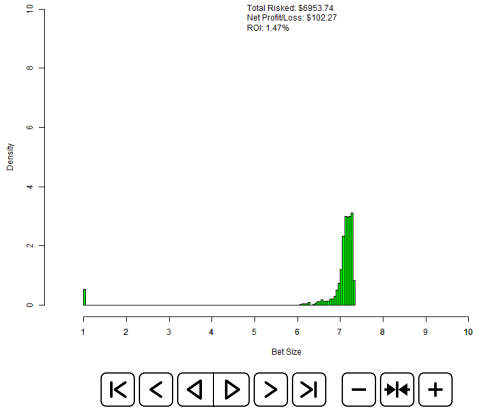
For Blackjack Problem III, the MCMC optimisation produces an agent that opts to stand on all hard totals but hit on all soft totals, whereas the Hybrid method selects to stand for both hard and soft totals. In both cases, neither method chooses to surrender or split pairs. Additionally, both methods yield betting agents whose bet-sizes are concentrated around a specific value of approximately 3 to 5 units per night. Despite these differences, both methods produce solutions which achieve comparable in-sample ROI %, suggesting the possibility that multiple combinations of decision-making strategies and bet-sizing schemes can lead to similar in-sample ROI %.

	MCMC			Hybrid		
	In-Sample	Out-of-Sample		In-Sample	Out-of-Sample	
Blackjack Problem	ROI %	$\mu_{ROI\%}$	$\sigma_{ROI\%}$	ROI %	$\mu_{ROI\%}$	$\sigma_{ROI\%}$
I	−14.8500	−16.0162	3.0645	-14.8500	−16.0162	3.0645
II	−3.9011	−0.6765	3.6044	-3.9154	-0.6841	3.6069
III	-11.9441	-15.7276	3.1629	-12.1605	-14.4514	2.8351

Table 6: ROI in-sample and out-of-sample performance for the three blackjack problems using $\beta = 50$ for Problem I, $\beta = 250$ for Problem II and $\beta = 50$ for Problem III.

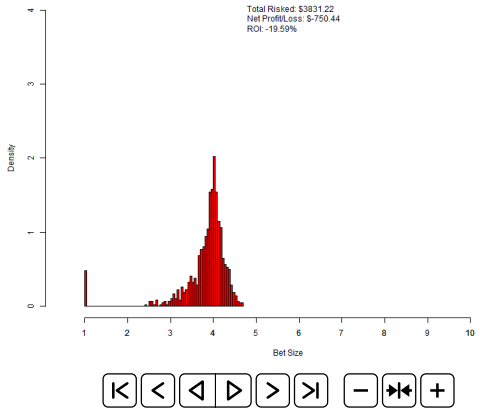


Bet-size ($\tilde{bet}_k(\hat{\boldsymbol{\theta}}^{\text{MCMC, (II)}})$) distributions per night according to solution derived from MCMC.

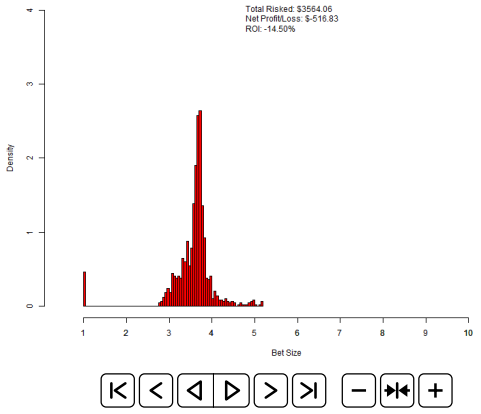


Bet-size ($\tilde{bet}_k(\hat{\boldsymbol{\theta}}^{\text{Hybrid, (II)}})$) distributions per night according to solution derived from GA Hybrid.

Figure 9: Bet-size distributions per night (over 100 nights) for Blackjack Problem II.



Bet-size ($\tilde{bet}_k(\hat{\boldsymbol{\theta}}^{\text{MCMC}})$) distributions per night according to solution derived from MCMC.



Bet-size ($\tilde{bet}_k(\hat{\boldsymbol{\theta}}^{\text{Hybrid}})$) distributions per night according to solution derived from GA Hybrid.

Figure 10: Bet-size distributions per night (over 100 nights) for Blackjack Problem III.

Furthermore, this application illustrates that, when the conditional distribution $p(\boldsymbol{\theta} \mid \sigma_\theta^2, \mathcal{D})$ is sufficiently sharp (achieved via an increased β), both the two-sample MCMC framework and the Hybrid method tend to optimise solutions which converge toward the same dominant mode of the conditional posterior: recalling from earlier that increasing β suppresses minor modes and accentuates dominant modes. This is congruent to what was said by with regards to SA; where a decreased temperature parameter (inverse to β) concentrates samples around global minima of the cost function. This results in comparable in-sample performance between the aforementioned methods. Such evidence supports the earlier postulate that an iterative optimisation procedure could serve as a viable replacement for the MH sampling in Block 1 of the two-sample MCMC framework in Section **??**, provided the posterior is sufficiently sharp.

4 Conclusion

The study aimed to illustrate the shortcomings of two-block MCMC, which is often employed to allow the training data to infer a level of regularization by incorporating the sampling of the dispersion parameter, σ_θ^2, into the algorithm. We showed that the pseudo-likelihood form, the likelihood sharpness parameter β, and the initial dispersion σ_{Init}^2 are in fact user-specified hyperparameters that exert a substantial influence on the degree of regularization inferred. As such, the use of a Bayesian hierarchical model

in this context does not genuinely infer regularization from the training data; rather, it is the *user* who determines the effective strength of regularization, albeit with additional steps.

Furthermore, we demonstrated that if one were to increase likelihood sharpness to an extreme, one may effectively reduce the two-block MCMC to a hybrid procedure in which the first block is replaced by a iterative optimisation procedure, yielding nearly identical in-sample performance to the original scheme. In this sense, the sampling of the dispersion parameter at each iteration functions primarily as a mechanism to add exploration to the search process, rather than as a means to infer regularization.

Future work should investigate treating the sharpness parameter in analogy to SA, whereby β is increased according to a cooling schedule within the two-block MCMC framework. However, we emphasize caution in this approach, as increasing sharpness may impair the mixing quality of the Markov chain.

Appendix A Neural Network Architecture

We define the feed-forward recursive relation in scalar form, hence the j^{th} node on the l^{th} layer for the i^{th} observation is given as:

$$a_j^l(i) = \sigma_l \left(\sum_{k=1}^{d_{l-1}} a_k^{l-1}(i) w_{kj}^l + b_j^l \right)$$

for $l = 1, 2, \dots L; j = 1, 2, \dots d_l; i = 1, 2 \dots N$. Now, $\sigma_l(.)$ denotes the activation function on layer l, d_{l-1} denotes the number of nodes in layer $l-1$, w_{kj}^l denotes the kj^{th} weight linking the k^{th} node layer $l-1$ and the j^{th} node in layer l with b_j^l denoting the j^{th} bias in layer l. The equation is evaluated subject to the initial conditions $a_j^{(0)} = x_{ij}$ for all j at the i^{th} training example.

Appendix B The True Count

In card counting (e.g., Hi-Lo system), the running count is the sum of values assigned to seen cards (e.g., +1 for 2-6, 0 for 7-9, -1 for 10-A). The true count adjusts this for the number of decks remaining. We define the running count at time k as $RC_k = \sum_{c \in \boldsymbol{\mathcal{H}}_k} \rho(c)$, where $\rho(c)$ is the count value of card c under the counting system. Assuming a $D_0 = 8$-deck shoe, the true count is defined as $TC_k = \frac{RC_k}{52 D_0 - |\boldsymbol{\mathcal{H}}_k|}$. A high true count implies a greater proportion of high cards (10s and Aces) remaining in the shoe, which statistically favors the player by increasing the likelihood of blackjack or beating the dealer, and improving the effectiveness of doubling and splitting actions.

Appendix C 3-Class Classification Particle Data

Table 7 provides an overview of the variables used in the particle classification task. The dataset consists of a two-dimensional spatial cross-section - represented by x and y coordinates - of three distinct sub-atomic particle types, with a total of 360 observations. The dataset is available in the project files, and the corresponding training and test set allocations are defined within the accompanying codebase.

Variable	Description
$X1$	First coordinate in cross-section
$Y2$	Second coordinate in cross-section
$Yi1$	Response: 1 if code-α, 0 otherwise.
$Yi2$	Response: 1 if code-β, 0 otherwise.
$Yi3$	Response: 1 if code-ρ, 0 otherwise.

Table 7: Variable descriptions for the particle classification dataset.