Supplementary Material: Confidence Sets and Hypothesis Testing in a Likelihood-Free Inference Setting

A. Algorithm for Simulating Labeled Sample for Estimating Odds Ratios

Algorithm 3 provides details on how to create the training sample $\mathcal{T}_B$ for estimating odds ratios. Out of the total training sample size B, a proportion p is generated by the stochastic forward simulator F_θ at different parameter values θ, while the remainder is sampled from a reference distribution G.

Algorithm 3 Generate a labeled sample of size B for estimating odds ratios

Require: stochastic forward simulator F_θ; reference distribution G, proposal distribution r_Θ over parameter space; training sample size B; parameter p of Bernoulli distribution

Ensure: labeled training sample

Draw parameter values $\theta_1, \ldots, \theta_B \overset{iid}{\sim} r_\Theta$
Assign labels $Y_1, \ldots, Y_B \overset{iid}{\sim} \sim Ber(p)$
for $i = 1, \ldots, B$ **do**
  if $Y_i == 1$ **then**
    Draw sample from forward simulator, $\mathbf{X}_i \sim F_\theta$
  end if
  if $Y_i == 0$ **then**
    Draw sample from reference distribution, $\mathbf{X}_i \sim G$
  end if
end for
return $\mathcal{T}_B = \{\theta_i, \mathbf{X}_i, Y_i\}_{i=1}^B$

B. Algorithm for Constructing Confidence Set for θ

Algorithm 4 summarizes the `ACORE` procedure for constructing confidence sets. First, we estimate parametrized odds to compute the `ACORE` test statistic τ (Eq. 4). Then, we compute a parametrized estimate of the critical values as a conditional distribution function of τ. Finally, we compute a confidence set for θ by the Neyman inversion technique (Neyman, 1937).

Algorithm 4 Construct confidence set for θ with coefficient $\gamma = 1 - \alpha$

Require: stochastic forward simulator F_θ; reference distribution G; proposal distribution r over Θ; parameter p of Bernoulli distribution; sample size B (for estimating odds ratios); sample size B' (for estimating critical values); probabilistic classifier; observed data $D = \{\mathbf{x}_1^{\text{obs}}, \ldots, \mathbf{x}_n^{\text{obs}}\}$; $\alpha \in (0, 1)$

Ensure: θ-values in confidence set

// Estimate odds ratios:
Generate labeled sample $\mathcal{T}_B$ according to Algorithm 3
Apply probabilistic classifier to $\mathcal{T}_B$ to learn class posterior probabilities, $\widehat{\mathbb{P}}(Y = 1|\theta, \mathbf{X})$, for all $\theta \in \Theta$ and $\mathbf{X} \in \mathcal{X}$
Let the estimated odds $\widehat{\mathbb{O}}(\theta, \mathbf{X}) = \frac{\widehat{\mathbb{P}}(Y=1|\theta,\mathbf{X})}{\widehat{\mathbb{P}}(Y=0|\theta,\mathbf{X})}$
Let the estimated odds ratios $\widehat{\mathbb{OR}}(\mathbf{X}; \theta_0, \theta_1) = \frac{\widehat{\mathbb{O}}(\theta_0,\mathbf{X})}{\widehat{\mathbb{O}}(\theta_1,\mathbf{X})}$, for all $\theta_1, \theta_2 \in \Theta$ and $\mathbf{X} \in \mathcal{X}$
// Estimate the critical value for a test δ_{θ_0} that rejects $\theta = \theta_0$ at significance level α:
Construct parametrized function $\widehat{C}_{\theta_0} := \widehat{F}^{-1}_{\tau|\theta_0}(\alpha|\theta_0)$ for $\theta_0 \in \Theta$ according to Algorithm 2
// Find parameter set for which the test δ_{θ_0} does not reject $\theta = \theta_0$:
`ThetaGrid` $\leftarrow$ grid of parameter values in Θ
$n_{\text{grid}} \leftarrow$ length(`ThetaGrid`)
Set $S \leftarrow \emptyset$
for `Theta0` $\in$ `ThetaGrid` **do**
  `Cutoff` $\leftarrow \widehat{F}^{-1}_{\tau|\theta_0}(\alpha \mid$ `Theta0`$)$
  `sumLogOR` $\leftarrow$ array with length n_{grid}
  for $j = 1, \ldots, n_{\text{grid}}$ **do**
    `Theta1` $\leftarrow$ `ThetaGrid`$[j]$
    `sumLogOR`$[j] \leftarrow \sum_{k=1}^n \log\left(\widehat{\mathbb{OR}}(\mathbf{x}_k^{\text{obs}}; \texttt{Theta0}, \texttt{Theta1})\right)$
  end for
  `TauObs` $\leftarrow \min($`sumLogOR`$)$
  if `TauObs` $>$ `Cutoff` **then**
    $S \leftarrow S \cup$ `Theta0`
  end if
end for
return S

C. Proofs

Proof of Proposition 3.1. If $\widehat{\mathbb{P}}(Y = 1|\theta, \mathbf{x}) = \mathbb{P}(Y = 1|\theta, \mathbf{x})$, then $\widehat{\mathbb{OR}}(\mathbf{x}; \theta_0, \theta_1) = \mathbb{OR}(\mathbf{x}; \theta_0, \theta_1)$. By Bayes rule and construction (Algorithm 3),

$$\mathbb{O}(\mathbf{x}; \theta) := \frac{\mathbb{P}(Y = 1|\theta, \mathbf{x})}{\mathbb{P}(Y = 0|\theta, \mathbf{x})} = \frac{f(\mathbf{x}|\theta)p}{g(\mathbf{x})(1-p)}.$$

Thus, the odds ratio at $\theta_0, \theta_1 \in \Theta$ is given by

$$\mathbb{OR}(\mathbf{x}; \theta_0, \theta_1) = \frac{f(\mathbf{x}|\theta_0)}{f(\mathbf{x}|\theta_1)},$$

and therefore

$$\begin{aligned}\tau(D;\Theta_0) &= \sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\left(\log\widehat{\mathbb{OR}}(\mathbf{x}_i^{\text{obs}};\theta_0,\theta_1)\right)\\ &= \sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\log\frac{f(\mathbf{x}_i^{\text{obs}}|\theta_0)}{f(\mathbf{x}_i^{\text{obs}}|\theta_1)}\\ &= \sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\log\left(\frac{\mathcal{L}(D;\theta_0)}{\mathcal{L}(D;\theta_1)}\right)\\ &= \Lambda(D;\Theta_0).\end{aligned}$$

□

Proof of Theorem 3.3. The union bound and Assumption 3.2 imply that

$$\sup_{\theta\in\Theta_0}\sup_{t\in\mathbb{R}}|\hat{F}_{B'}(t|\theta)-F(t|\theta)|\xrightarrow[B'\longrightarrow\infty]{\mathbb{P}}0.$$

It follows that

$$\sup_{\theta\in\Theta_0}|\hat{F}^{-1}_{B'}(\alpha|\theta)-F^{-1}(\alpha|\theta)|\xrightarrow[B'\longrightarrow\infty]{\mathbb{P}}0.$$

The result follows from the fact that

$$\begin{aligned}0\leq|C_{B,B'}-C^*_B| &= |\sup_{\theta\in\Theta_0}\hat{F}^{-1}_{B'}(\alpha|\theta)-\sup_{\theta\in\Theta_0}F^{-1}(\alpha|\theta)|\\ &\leq \sup_{\theta\in\Theta_0}|\hat{F}^{-1}_{B'}(\alpha|\theta)-F^{-1}(\alpha|\theta)|,\end{aligned}$$

and thus

$$|C_{B,B'}-C^*_B|\xrightarrow[B'\longrightarrow\infty]{\mathbb{P}}0.$$

□

Lemma C.1. *If* $(\widehat{\mathbb{P}}(Y=1|\theta,\mathbf{X}))_{\theta\in\Theta}\xrightarrow[B\longrightarrow\infty]{\mathbb{P}}(\mathbb{P}(Y=1|\theta,\mathbf{X}))_{\theta\in\Theta}$ *and* $|\Theta|<\infty$*, then*

$$\tau(\mathcal{D};\Theta_0)\xrightarrow[B\longrightarrow\infty]{\mathbb{P}}\sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\log\left(\mathbb{OR}(\mathbf{X}_i^{obs};\theta_0,\theta_1)\right)$$

Proof. For every $\theta_0,\theta_1\in\Theta$, it follows directly from the properties of convergence in probability that

$$\begin{aligned}&\sum_{i=1}^{n}\log\left(\widehat{\mathbb{OR}}(\mathbf{X}_i^{\text{obs}};\theta_0,\theta_1)\right)\\ &\qquad\xrightarrow[B\longrightarrow\infty]{\mathbb{P}}\sum_{i=1}^{n}\log\left(\mathbb{OR}(\mathbf{X}_i^{\text{obs}};\theta_0,\theta_1)\right)\end{aligned}$$

The conclusion of the lemma follows from the continuous mapping theorem. □

Proof of Theorem 3.4. Lemma C.1 implies that $\tau_B(\mathcal{D};\Theta_0)$ converges in distribution to $\sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\log\left(\mathbb{OR}(\mathbf{X}_i^{\text{obs}};\theta_0,\theta_1)\right)$. Now, from Slutsky's theorem,

$$\begin{aligned}&\tau_B(\mathcal{D};\Theta_0)-\widehat{C}_B\\ &\qquad\xrightarrow[B\longrightarrow\infty]{\text{Dist}}\sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\log\left(\mathbb{OR}(\mathbf{X}_i^{\text{obs}};\theta_0,\theta_1)\right)-C^*.\end{aligned}$$

It follows that

$$\begin{aligned}&\mathbb{P}\left(\widehat{\phi}_{B,\widehat{C}_B}(\mathcal{D})=1|\theta\right)=\mathbb{P}\left(\tau_B(\mathcal{D};\Theta_0)-\widehat{C}_B\leq 0|\theta\right)\\ &\xrightarrow[B\longrightarrow\infty]{}\mathbb{P}\Big(\sup_{\theta_0\in\Theta_0}\inf_{\theta_1\in\Theta}\sum_{i=1}^{n}\log\left(\mathbb{OR}(\mathbf{X}_i^{\text{obs}};\theta_0,\theta_1)\right)\\ &\qquad\qquad-C^*\leq 0|\theta\Big)\\ &=\mathbb{P}\left(\phi_{C^*}(\mathcal{D})=1|\theta\right),\end{aligned}$$

where the last equality follows from Proposition 3.1.

□

D. Toy Examples

This section provides details on the toy examples of Section 4. We use the `sklearn` ecosystem (Pedregosa et al., 2011) implementation of the following probabilistic classifiers:

- multi-layer perceptron (MLP) with default parameters, but no L^2 regularization ($\alpha=0$);
- quadratic discriminant analysis (QDA) with default parameters;
- nearest neighbors (NN) classifier, with number of neighbors equal to the rounded square root of the number of data points available (as per Duda et al. (2001)).

Table 3 reports the observed coverage for the settings of Tables 1 and 2. Critical values or C are estimated with quantile gradient boosted trees (100 trees with maximum depth equal to 3), a training sample size $B'=5000$, observed data D of sample size $n=10$, nominal coverage of 90%, and averaging over 100 repetitions. The table shows that we for all cases achieve results in line with the nominal confidence level.[5]

Our goodness-of-fit procedure shown in Figure 3 uses a set $\mathcal{T}_{B''}$ with size $B''=250$ (as defined in Section 3.3); Figure 5 shows the goodness-of-fit plot for the Gaussian mixture model example, where the coverage is estimated via logistic regression and the critical values are estimated via quantile gradient boosted trees. For the Poisson example

[5]The 95% CI of a binomial distribution with probability $p=0.9$ over 100 repetitions is in fact $[0.84,0.95]$. This interval includes the observed coverages listed in Table 3.

B	*Classifier*	*Poisson Example Coverage*	*GMM Example Coverage*
100	MLP	0.91	0.87
	NN	0.91	0.91
	QDA	0.90	0.88
500	MLP	0.91	0.91
	NN	0.93	0.95
	QDA	0.94	0.92
1000	MLP	0.91	0.92
	NN	0.89	0.88
	QDA	0.91	0.93

Table 3. Observed coverage of the toy examples in Tables 1 and 2. These values are consistent with what we would expect for 100 trials with a nominal confidence level of 90%; see text.

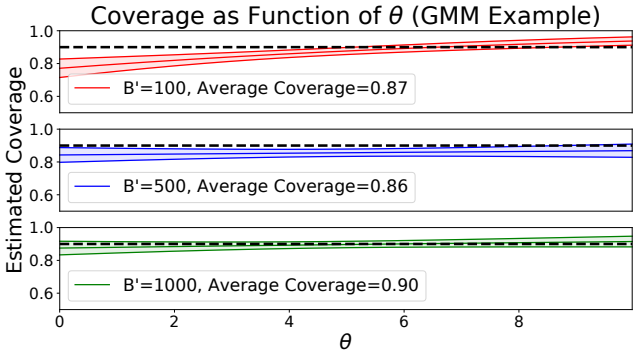

Figure 5. Estimated coverage as a function of θ in the Gaussian mixture model example for ACORE with different values of B'. Logistic regression is used to estimate mean coverage and one standard deviation prediction bands. (We here use $n = 10$, a MLP classifier with $B = 1000$ and quantile gradient boosted trees).

a training sample size of $B' = 500$ seems to be enough to achieve correct coverage, whereas the Gaussian mixture model example requires $B' = 1000$.

Next we compare our goodness-of-fit diagnostic with diagnostics obtained via standard Monte Carlo sampling. Figure 6 shows the MC coverage as a function of θ for the Poisson example (left) and the Gaussian mixture model example (right). In both cases 100 MC samples are drawn at 100 parameter values chosen uniformly. The empirical ACORE coverage is computed over the MC samples at each chosen θ. This MC procedure is expensive: it uses a total of $10,000$ simulations, which is 40 times the number used in our goodness-of-fit procedure. The observed coverage of the Poisson example (Figure 6, left) indicates that $B' = 500$ is sufficient to achieve the nominal coverage of 90%. For the Gaussian mixture model example (Figure 6, right), we detect undercoverage for very small values of θ. This discrepancy is due to the fact that, at $\theta = 0$, the mixture collapses into a single Gaussian, structurally different from the GMM at any other $\theta > 0$ and closer to the $\mathcal{N}(0, 5^2)$ reference distribution.

Our goodness-of-fit procedure is able to identify that the actual coverage is far from the nominal coverage at small values of θ, when the training sample size B' for estimating C is too small. More specifically, Figure 5 shows a noticeable tilt in the prediction bands for $B' = 100$ and 500. However, as B' increases, the estimation of critical values becomes more precise and the estimated confidence intervals pass our goodness-of-fit diagnostic at, for example, $B' = 1000$. Future studies will provide a more detailed account on how such boundary effects depend on the method for estimating the coverage.

E. Signal Detection in High Energy Physics

Here we consider the signal detection example in Section 5 We describe the details of the construction of ACORE confidence sets which used the strategy in Section 5 to choose ACORE components and parameters. For learning the odds ratio, we compared the following classifiers:

- logistic regression,
- quadratic discimininant analysis (QDA) classifier,
- nearest neighbor classifier,
- gradient boosted trees using $\{100, 500, 1000\}$ trees with maximum depth $\{3, 5, 10\}$,
- Gaussian process classifiers[6] with radial basis functions kernels with variance $\{1, .5, .1\}$,
- feed-forward deep neural networks, with $2, ..., 6$ deep layers, number of neurons between $2^{\{4,...,10\}}$ and either ReLu or hyperbolic tangent activations.

For estimating the critical values, we considered the following quantile regression algorithms:

- gradient boosted trees using $\{100, 250, 500\}$ trees with maximum depth $\{3, 5, 10\}$,
- random forest quantile regression with $\{100, 250, 500\}$ trees,
- deep quantile regression with $\{2, 3\}$ deep layers, $2^{\{4,..,6\}}$ neurons and ReLu activations (using the PyTorch implementation (Paszke et al., 2019)).

All computations were performed on 8-Core Intel Xeon CPUs X5680 at 3.33GHz.

[6]GP classifiers were used only with sample sizes B below $10,000$, as the matrix inversion quickly becomes computationally infeasible for larger values of B.

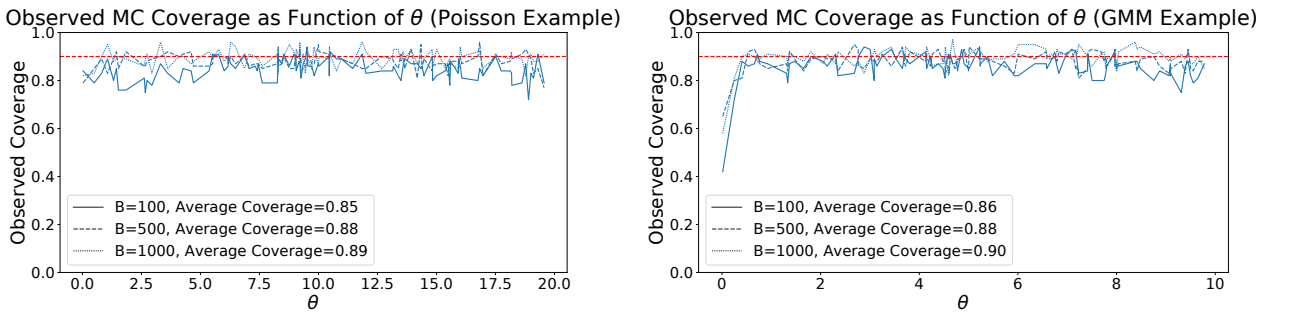

Figure 6. Observed ACORE coverage across the parameter space for the Poisson example (left) and the Gaussian mixture model example (right). The coverage is computed with Monte Carlo samples of size 100, each sampled at a θ chosen uniformly over the parameter space. Odds ratios are computed with a QDA classifier for the Poisson example, and an MLP classifier for the GMM example (as in Figures 3 and 5). We observe undercoverage at small θ for the GMM (right) due to the mixture collapsing into a single Gaussian as $\theta \to 0$.

Figure 7 illustrates the two steps in identifying the four components of ACORE. We first use a validation set of $5,000$ simulations to determine which probabilistic classifier and training sample size B minimize the cross entropy loss. Figure 7 (left) shows the cross entropy loss of the best four classifiers as function of B. The minimum is achieved by a 5-layer deep neural network (DNN) at $B = 100,000$ with a cross entropy loss of 58.509×10^{-2}, closely followed by QDA with 58.512×10^{-2} at $B = 50,000$. Given how similar the loss values are, we select both classifiers to follow-up on. In Figure 7 (right), the "estimated correct coverage" represents the proportion of the parameter space that passes our diagnostic procedure. The lowest B' with correct coverage is achieved by the five-layer DNN classifier (for estimating odds ratios) at $B' = 25,000$ with critical values estimated via a two-layer deep quantile regression algorithm. None of the quantile regression algorithms pass a diagnostic test with a nominal coverage of 90% at the one standard deviation level when using the QDA classifier. We therefore do not use QDA in Section 5.

Based on the analysis above, we choose the following ACORE components: (i) a five-layer DNN for learning odds ratios, (ii) $B = 100,000$, (iii) a two-layer deep quantile regression for estimating critical values, and (iv) $B' = 25,000$. Figure 4 shows the confidence sets computed with this choice.

F. Cross Entropy Loss Analysis

In this work, we use the cross entropy loss to measure the accuracy of the probabilistic predictions of the classifier. That is, we calibrate the estimated odds function $g(\theta, \mathbf{x}) := \widehat{\mathbb{P}}(Y = 1|\theta, \mathbf{x})/\widehat{\mathbb{P}}(Y = 0|\theta, \mathbf{x})$ as follows: Consider a sample point $\{\theta, \mathbf{x}, y\}$ generated according to Algorithm 3. Let p be a Ber(y) distribution, and q be a Ber $\left(\widehat{\mathbb{P}}(Y = 1|\theta, \mathbf{x})\right) = \text{Ber}\left(\frac{g(\theta,\mathbf{x})}{1+g(\theta,\mathbf{x})}\right)$ distribution. The *cross entropy* between p and q is given by

$$\begin{aligned} L_{\text{CE}}(g; \{\theta, \mathbf{x}, y\}) &= -y \log\left(\frac{g(\theta, \mathbf{x})}{1 + g(\theta, \mathbf{x})}\right) \\ &\quad - (1 - y) \log\left(\frac{1}{1 + g(\theta, \mathbf{x})}\right) \\ &= -y \log\left(g(\theta, \mathbf{x})\right) + \log\left(1 + g(\theta, \mathbf{x})\right). \end{aligned}$$

For every $\mathbf{x}$ and θ, the expected cross entropy $\mathbb{E}[L_{\text{CE}}(g; \{\theta, \mathbf{x}, Y\})]$ is minimized by $g(\theta, \mathbf{x}) = \mathbb{O}(\theta, \mathbf{x})$. Thus we can measure the performance of an estimator g of the odds by the risk

$$R_{\text{CE}}(g) = \mathbb{E}[L_{\text{CE}}(g; \{\theta, \mathbf{X}, Y\})].$$

The cross entropy loss is not the only loss function that is minimized by the true odds function, but it is usually easy to compute in practice. It is also well known that minimizing the cross entropy loss between the estimated distribution q and the true distribution p during training is equivalent to minimizing the Kullback-Leibler (KL) divergence between the two distributions, as

$$KL(p||q) = H(p, q) - H(p),$$

where $H(p, q)$ is the cross entropy and $H(p)$ is the entropy of the true distribution. By Gibbs' inequality (MacKay, 2002), we have that $KL(p||q) \geq 0$; hence the entropy $H(p)$ of the true distribution lower bounds the cross entropy with the minimum achieved when $p = q$. Hence, we can connect the cross entropy loss to the ACORE statistic.

Proposition F.1. *If the probabilistic classifier in* ACORE *achieves the minimum of the cross entropy loss, then the constructed* ACORE *statistic* (4) *is equal to the likelihood ratio statistic* (3).

Proof of Prop F.1. The proof follows from Proposition 3.1 and the expected cross entropy loss is minimized if and only if $\widehat{\mathbb{O}}(\theta, \mathbf{x}) = \mathbb{O}(\theta, \mathbf{x})$. □

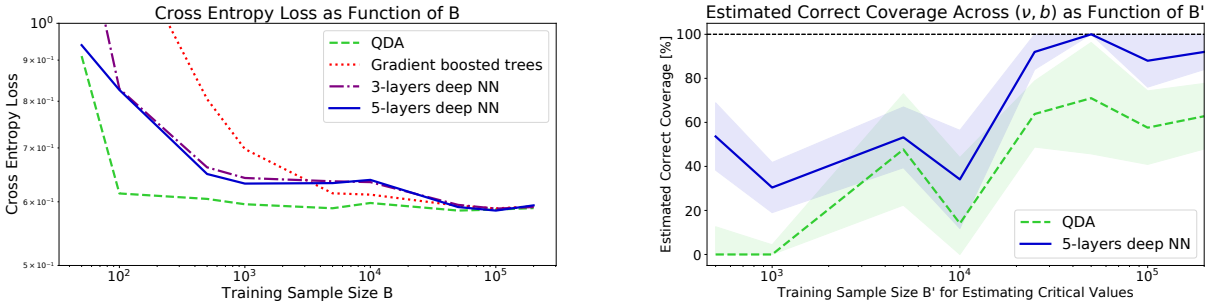

Figure 7. Using the strategy in Section 5 to choose ACORE components for the signal detection example. *Left:* The cross entropy loss of the best four classifiers, shown as a function of B. In order of increasing loss: 5-layer DNN ([512, 256, 64, 32, 32] neurons, ReLu activations), QDA classifier, 3-layer DNN ([64, 32, 32] neurons, ReLu activations) and gradient boosted trees (1000 trees with maximum depth 5). Because the first two classifiers (the 5-layer DNN and QDA) achieve a very similar minimum loss, we consider both classifiers in the follow-up step. *Right*: Proportion of the (ν, b) parameter space where the best two classifiers pass our goodness-of-fit procedure with a nominal coverage of 90%. Both the mean value curves and the $\pm$ one standard deviation prediction bands are computed via logistic regression. Critical values are estimated via a two-layer deep quantile regression ([64,64] neurons, ReLu activations), which passed the diagnostic at the lowest sample size ($B' = 25,000$, with the 5-layers DNN). Based on the results, we choose the 5-layer DNN with $B' = 25,000$.

In addition, we show that the convergence of the class posterior implies the convergence of the cross entropy to the entropy of the true distribution. This supports our decision to use the cross entropy loss when selecting the probabilistic classifier and sample size B.

Lemma F.2. *If for every* $\theta \in \Theta$

$$q := \widehat{\mathbb{P}}(Y = 1|\theta, \mathbf{X}) \xrightarrow[B\longrightarrow\infty]{\mathbb{P}} p := \mathbb{P}(Y = 1|\theta, \mathbf{X}),$$

then $H(p, q) \xrightarrow[B\longrightarrow\infty]{\mathbb{P}} H(p)$.

Proof of Lemma F.2. We can rewrite the cross entropy $H(p, q)$ and entropy $H(p)$ as

$$H(p, q) = -\sum_{y\in\{0,1\}} \int_{\mathcal{X}\times\Theta} p \log(q)\, \mathrm{d}\mathbb{P}(\mathbf{x}, \theta),$$

$$H(p) = -\sum_{y\in\{0,1\}} \int_{\mathcal{X}\times\Theta} p \log(p)\, \mathrm{d}\mathbb{P}(\mathbf{x}, \theta).$$

In addition, for any $(\mathbf{X}, \theta)$, it also holds that $|q| \leq 1$. The lemma follows by combining the dominated convergence theorem with the continuous mapping theorem for the logarithm. □

G. Comparison with Monte Carlo Synthetic Likelihood-Based Methods

In this section we compare the performance of ACORE with Monte-Carlo (MC) synthetic likelihood-based methods, more specifically Gaussian process (GP) interpolation (Frate et al., 2017). The latter method first simulates multiple sample points for a few different values of θ. For each fixed θ, one fits a Gaussian synthetic likelihood function. The GP likelihood model is then used to smoothly interpolate across the parameter space by fitting a mean function $m(\theta)$ and a covariance function $\Sigma(\theta)$. As a note, Cranmer et al. (2019) point out that such MC methods are less efficient than methods that estimate the likelihood ratio directly because of the need to first estimate the entire likelihood.

For our comparison, we use the two toy examples described in Section 4 and Table 1. To allocate B sample points for the GP interpolation, we use the following strategy: For $q \in \{5, 10, 25\}$, first choose $\theta_1, ..., \theta_q$ on an evenly spaced grid across the parameter space. Then, generate $N = B/q$ sample points $\mathbf{X}_1, ..., \mathbf{X}_N$ at each location θ.

Table 4 summarizes the results. Unlike Table 2, we do not report the cross-entropy loss because GP interpolation is not a classification algorithm; instead we report the mean squared error in estimating the likelihood ratio across the parameter space. Our results show that when the simulated data at each θ are approximately Gaussian, as in the Poisson example, MC-based GP interpolation provides a better approximation of the likelihood ratio due to its parametric assumptions. However, when the parametric assumptions are not valid, as in the GMM example, MC-based GP fails to approximate the likelihood ratio regardless of how large N or B are. In such settings, we do better with a fully nonparametric approach. As a note, MC-based GP uses the asymptotic χ^2 approximation by Wilks' theorem to determine the critical values of the confidence sets. In our experiments, using quantile regression for critical values instead (as in ACORE) led to a significant increase in power for the GP likelihood models: from ≈ 0.48 to ≈ 0.51 for the Poisson example, and from ≈ 0.02 to ≈ 0.2 for the GMM example.

Poisson Example

B	*Classifier*	*90 % Mean Squared Error Interval*	*Average Power*	*Size of Confidence Set [%]*
100	MLP	[2.14, 989.78]	0.27	72.8 ± 16.4
	NN	[4.14, 4074.65]	0.25	75.6 ± 23.2
	QDA	[0.41, 34.79]	0.41	$\mathbf{60.1 \pm 14.9}$
	G.P. (5)	**[0.05, 4.09]**	0.47	$\mathbf{53.5 \pm 9.2}$
	G.P. (10)	**[0.06, 4.97]**	**0.48**	$\mathbf{53.2 \pm 10.7}$
	G.P. (25)	**[0.03, 6.54]**	**0.48**	$\mathbf{53.2 \pm 10.8}$
500	MLP	[0.86, 22.45]	0.38	62.2 ± 19.1
	NN	[1.95, 32.78]	0.37	$\mathbf{64.2 \pm 17.3}$
	QDA	[0.08, 6.95]	0.45	$\mathbf{55.5 \pm 10.8}$
	G.P. (5)	**[0.01, 0.81]**	**0.49**	$\mathbf{52.4 \pm 5.6}$
	G.P. (10)	**[0.02, 0.85]**	**0.49**	$\mathbf{52.0 \pm 5.4}$
	G.P. (25)	**[0.01, 1.12]**	**0.48**	$\mathbf{52.5 \pm 6.0}$
1,000	MLP	[0.81, 21.44]	0.42	$\mathbf{58.8 \pm 17.0}$
	NN	[1.77, 17.88]	0.45	$\mathbf{56.1 \pm 16.2}$
	QDA	[0.06, 2.83]	**0.49**	$\mathbf{52.1 \pm 9.0}$
	G.P. (5)	**[0.01, 0.48]**	**0.49**	$\mathbf{52.3 \pm 5.0}$
	G.P. (10)	**[0.01, 0.46]**	**0.48**	$\mathbf{52.5 \pm 5.3}$
	G.P. (25)	**[0.01, 0.45]**	**0.48**	$\mathbf{52.6 \pm 5.5}$
-	**Exact**	-	**0.54**	$\mathbf{45.0 \pm 4.9}$

GMM Example

B	*Classifier*	*90 % Mean Squared Error Interval* ($\times 10^3$)	*Average Power*	*Size of Confidence Set [%]*
100	MLP	**[0.34, 1.46]**	**0.87**	$\mathbf{14.5 \pm 4.5}$
	NN	[1.33, 11.77]	0.49	52.1 ± 24.7
	QDA	[2.88, 3.56]	0.16	84.0 ± 21.8
	G.P. (5)	[3.35, 3.82]	0.02	97.7 ± 8.8
	G.P. (10)	[3.34, 3.82]	0.03	96.9 ± 9.5
	G.P. (25)	[3.36, 3.82]	0.02	98.2 ± 6.1
500	MLP	**[0.44, 1.35]**	**0.90**	12.1 ± 2.8
	NN	[0.99, 2.65]	0.57	44.0 ± 23.3
	QDA	[3.14, 3.73]	0.16	83.8 ± 22.2
	G.P. (5)	[3.39, 3.83]	0.00	100.0 ± 0.0
	G.P. (10)	[3.39, 3.83]	0.01	99.1 ± 5.5
	G.P. (25)	[3.38, 3.83]	0.00	99.8 ± 1.5
1,000	MLP	**[0.53, 1.17]**	**0.90**	$\mathbf{12.1 \pm 2.8}$
	NN	[0.57, 2.04]	0.71	30.2 ± 18.5
	QDA	[3.26, 3.94]	0.14	85.7 ± 20.1
	G.P. (5)	[3.39, 3.98]	0.00	100.0 ± 0.0
	G.P. (10)	[3.39, 3.98]	0.00	100.0 ± 0.0
	G.P. (25)	[3.39, 3.98]	0.00	99.9 ± 1.2
-	**Exact**	-	**0.92**	$\mathbf{9.5 \pm 2.0}$

Table 4. Results for ACORE (MLP, NN, QDA) and Gaussian Process interpolation (GP for $q = 5, 10, 25$; see text) for the two toy examples, Poisson example (left) and GMM example (right), of Section 4. The tables list the mean squared error (MSE) between the estimated and true likelihood, the power (averaged over θ) and the size of confidence sets, for different values of B and for different classifiers. We report a 90% confidence interval for the MSE, together with the mean and standard deviation of the size of the estimated 90% confidence set for θ. Best results for each training sample size B are marked in bold-faced. All fitted classifiers produce valid 90% confidence sets for θ according to our diagnostics.

H. Comparison with Calibrated Approximate Ratio of Likelihood Classifiers

In this section we compare the performance of ACORE with the calibrated approximate ratio of likelihood (CARL) estimator by Cranmer et al. (2015). CARL approximates the likelihood ratio $\Lambda(D; \Theta_0) = \mathcal{L}(D; \theta_0)/\mathcal{L}(D; \theta_1)$ by turning the density ratio estimation into a supervised classification problem, where a probabilistic classifier is trained to separate samples from F_{θ_0} and F_{θ_1}. As such, CARL classifiers are "doubly parameterized" by θ_0 and θ_1, whereas the ACORE classifier is parameterized by a *single* parameter θ in the definition of the odds of F_θ versus G.

In our study, we include three different CARL classifiers, implemented with the MADMINER neural network-based software (Brehmer et al., 2020a; 2019): (a) a shallow perceptron with 100 neurons (equivalent to the MLP used in Section 4), (b) a 2-layer deep network with 20 neurons per layer, and (c) a 2-layer deep network with 20 and 50 neurons in the two layers respectively.[7] To allocate B sample points for interpolation we devised two schemes: (i) a uniform sampling, and (ii) a Monte Carlo sampling over the parameter space. For (i), we uniformly sample B parameters and then generate a sample point $\mathbf{X}$ at each parameter value. For (ii), we first select evenly spaced parameters $\theta_{0,1}, ..., \theta_{0,q}$ and $\theta_{1,1} ..., \theta_{1,q}$, for the numerator and the denominator respectively. We set $q \in \{10, 20, 30\}$, resulting in $N = B/q$ sample points $\mathbf{X}_1, ..., \mathbf{X}_N$ at each θ location. Because the χ^2 approximation by Wilks' theorem did not yield valid confidence sets for CARL classifiers, we computed critical values as in ACORE Algorithm 1.

Table 5 shows the results of ACORE and CARL for the synthetic data in Section 4 and Table 1. For both the Poisson and GMM examples, CARL classifiers yield a higher mean squared error in estimating the likelihood ratio, as well as lower power and larger confidence intervals.

I. Runtime Analysis

In this section we provide a runtime analysis for constructing one ACORE confidence set for the two examples in Section 4 and Table 2. We also provide a running time comparison with the two methods described in Sections G and H. This analysis was performed on a 8-Core Intel Xeon 3.33GHz X5680 CPU.

The procedure for constructing confidence sets with ACORE is outlined in Algorithm 4. In this analysis we break the computation into 4 steps: (i) odds ratio training as described by Algorithm 3, (ii) computing the test statistic (4) for the observed data, (iii) computing the test statistic (4) in the B' sample as described by Algorithm 2 and (iv) quantile regression algorithm training. Table 6 summarizes our running

[7] Changing the number of neurons per layers did not seem to provide a significant difference in performance for the 2-layer deep networks. Number of epochs and learning rate were manually tuned (with a search in the range $[20, 200]$ and $10^{\{-6,\dots,-2\}}$ respectively).

Poisson Example

B	*Classifier*	*90 % Mean Squared Error Interval*	*Average Power*	*Size of Confidence Set [%]*
200	MLP	[3.25, 1305.45]	0.17	82.7 ± 15.0
	NN	[2.88, 185.47]	0.34	66.9 ± 20.7
	QDA	**[0.20, 25.16]**	**0.45**	$\mathbf{55.8 \pm 13.2}$
	MLP (MC)	[2.51, 38.10]	0.24	76.1 ± 21.3
	(20,20) DNN (MC)	[2.53, 25.41]	0.19	80.9 ± 17.8
	(50,20) DNN (MC)	[2.76, 26.00]	0.19	81.3 ± 17.8
	MLP (U)	[2.03, 45.19]	0.19	81.3 ± 19.2
	(20,20) DNN (U)	[2.95, 19.76]	0.24	76.6 ± 19.8
	(50,20) DNN (U)	[2.43, 18.72]	0.23	77.8 ± 20.1
800	MLP	[1.69, 450.81]	0.27	73.0 ± 20.1
	NN	[1.47, 19.32]	0.42	$\mathbf{59.2 \pm 15.9}$
	QDA	**[0.04, 5.03]**	**0.49**	$\mathbf{52.0 \pm 9.3}$
	MLP (MC)	[2.38, 24.50]	0.22	78.5 ± 21.0
	(20,20) DNN (MC)	[2.49, 21.49]	0.25	75.3 ± 18.8
	(50,20) DNN (MC)	[2.52, 18.13]	0.23	76.9 ± 20.1
	MLP (U)	[2.04, 23.24]	0.20	79.9 ± 17.4
	(20,20) DNN (U)	[2.48, 17.36]	0.22	77.9 ± 17.6
	(50,20) DNN (U)	[2.25, 17.87]	0.21	78.9 ± 20.0
1,800	MLP	[0.81, 19.11]	0.37	$\mathbf{63.7 \pm 21.1}$
	NN	[1.09, 11.27]	0.44	$\mathbf{56.9 \pm 14.3}$
	QDA	**[0.03, 1.60]**	**0.50**	$\mathbf{51.0 \pm 6.6}$
	MLP (MC)	[2.13, 35.39]	0.18	82.4 ± 17.7
	(20,20) DNN (MC)	[2.74, 28.15]	0.20	80.3 ± 19.7
	(50,20) DNN (MC)	[2.62, 28.15]	0.18	81.9 ± 19.5
	MLP (U)	[2.15, 25.51]	0.19	81.4 ± 19.8
	(20,20) DNN (U)	[2.34, 15.93]	0.23	77.0 ± 22.6
	(50,20) DNN (U)	[2.38, 17.97]	0.19	81.6 ± 17.2
-	**Exact**	-	**0.54**	$\mathbf{45.0 \pm 4.9}$

GMM Example

B	*Classifier*	*90 % Mean Squared Error Interval* ($\times 10^3$)	*Average Power*	*Size of Confidence Set [%]*
200	MLP	**[0.56, 1.69]**	**0.88**	$\mathbf{14.2 \pm 8.2}$
	NN	[1.13, 4.17]	0.50	51.5 ± 24.8
	QDA	[3.05, 3.63]	0.12	87.6 ± 19.7
	MLP (MC)	[3.03, 3.61]	0.27	73.5 ± 20.5
	(20,20) DNN (MC)	[3.13, 3.70]	0.25	75.6 ± 20.0
	(50,20) DNN (MC)	[3.16, 3.67]	0.28	72.8 ± 19.6
	MLP (U)	[3.01, 3.72]	0.30	70.2 ± 21.2
	(20,20) DNN (U)	[3.18, 3.87]	0.24	76.3 ± 21.5
	(50,20) DNN (U)	[3.12, 3.92]	0.27	73.0 ± 21.2
800	MLP	**[0.89, 1.59]**	**0.90**	$\mathbf{12.1 \pm 2.5}$
	NN	[0.78, 2.31]	0.69	32.0 ± 18.9
	QDA	[3.23, 3.66]	0.14	86.1 ± 20.4
	MLP (MC)	[3.02, 3.58]	0.30	70.8 ± 20.4
	(20,20) DNN (MC)	[3.10, 3.63]	0.27	73.6 ± 20.2
	(50,20) DNN (MC)	[3.03, 3.47]	0.30	70.5 ± 18.5
	MLP (U)	[3.01, 3.62]	0.26	74.7 ± 20.6
	(20,20) DNN (U)	[3.12, 3.64]	0.26	74.4 ± 19.2
	(50,20) DNN (U)	[3.00, 3.56]	0.29	71.8 ± 19.9
1,800	MLP	**[0.33, 1.55]**	**0.90**	$\mathbf{11.5 \pm 2.6}$
	NN	**[0.32, 1.57]**	0.83	$\mathbf{19.3 \pm 10.3}$
	QDA	[3.29, 3.81]	0.16	83.7 ± 22.2
	MLP (MC)	[2.99, 3.54]	0.33	67.5 ± 19.6
	(20,20) DNN (MC)	[3.02, 3.54]	0.31	69.7 ± 19.3
	(50,20) DNN (MC)	[2.95, 3.51]	0.38	63.1 ± 15.9
	MLP (U)	[2.99, 3.45]	0.33	67.7 ± 17.0
	(20,20) DNN (U)	[3.02, 3.56]	0.33	67.3 ± 18.0
	(50,20) DNN (U)	[2.98, 3.41]	0.38	63.1 ± 15.3
-	**Exact**	-	**0.92**	$\mathbf{9.5 \pm 2.0}$

Table 5. Results for ACORE (MLP, NN, QDA) and CARL or uniform (U) and Monte-Carlo (MC) sampling schemes in the Poisson example (left) and GMM example (right) settings of Section 4. The tables list the mean squared error (MSE) between the estimated and true likelihood, the power (averaged over θ) and the size of confidence sets, for different values of B and for different classifiers. We report a 90% confidence interval for the MSE, together with the mean and standard deviation of the size of the estimated 90% confidence set for θ. The best results for each training sample size B are marked in bold-faced.

times results. ACORE constructs one confidence set in less than 20 and 30 seconds for Poisson and GMM examples respectively. The main computational bottleneck is step (iii), while the computation time of step (i) increases with the sample size B.

Figure 8 shows the results of comparing confidence set construction runtimes with MC GP and CARL classifiers. For both the Poisson and the GMM examples, we only consider the best performing ACORE classifiers, and the two CARL classifiers with 20 hidden units in both layers. Results show ACORE classifiers are comparable with GP interpolation in terms of running times, while CARL classifiers tend to have significantly longer runtimes.

Running Times to Generate a Confidence Set (Seconds) – Poisson Example

B	*Classifier*	*Odds Ratio Training*	*Odds Ratio Prediction*	*Calculate (4) for B' Samples*	*Quantile Regression Training*	*Total Running Time*
100	MLP	0.38 ± 0.31	0.42 ± 0.10	10.40 ± 0.71	0.66 ± 0.28	11.86 ± 1.02
	NN	0.03 ± 0.01	0.35 ± 0.12	9.83 ± 4.99	0.82 ± 0.67	11.02 ± 5.73
	QDA	0.02 ± 0.01	0.18 ± 0.11	4.50 ± 2.65	0.58 ± 0.21	5.29 ± 2.96
500	MLP	1.62 ± 0.39	0.46 ± 0.04	11.49 ± 0.45	0.68 ± 0.09	14.26 ± 0.61
	NN	0.13 ± 0.01	0.54 ± 0.03	13.28 ± 0.26	0.66 ± 0.04	14.60 ± 0.29
	QDA	0.13 ± 0.01	0.16 ± 0.01	4.12 ± 0.09	0.65 ± 0.06	5.05 ± 0.14
1,000	MLP	2.65 ± 0.88	0.48 ± 0.08	11.93 ± 1.93	0.73 ± 0.06	15.79 ± 2.30
	NN	0.24 ± 0.04	0.77 ± 0.21	17.90 ± 2.82	0.67 ± 0.10	19.59 ± 2.83
	QDA	0.27 ± 0.08	0.17 ± 0.05	4.37 ± 1.02	0.64 ± 0.16	5.45 ± 1.29

Running Times to Generate a Confidence Set (Seconds) – GMM Example

B	*Classifier*	*Odds Ratio Training*	*Odds Ratio Prediction*	*Calculate (4) for B' Samples*	*Quantile Regression Training*	*Total Running Time*
100	MLP	5.89 ± 1.66	0.45 ± 0.18	10.79 ± 2.06	0.60 ± 0.21	17.74 ± 3.92
	NN	0.03 ± 0.00	0.29 ± 0.06	8.60 ± 2.84	0.61 ± 0.18	9.53 ± 3.05
	QDA	0.03 ± 0.01	0.14 ± 0.04	3.81 ± 1.38	0.52 ± 0.14	4.50 ± 1.57
500	MLP	9.89 ± 1.34	0.43 ± 0.06	11.64 ± 0.64	0.69 ± 0.06	22.64 ± 1.83
	NN	0.17 ± 0.01	0.52 ± 0.04	13.11 ± 0.79	0.63 ± 0.07	14.43 ± 0.85
	QDA	0.16 ± 0.01	0.15 ± 0.02	4.05 ± 0.26	0.59 ± 0.08	4.94 ± 0.35
1,000	MLP	13.40 ± 2.60	0.47 ± 0.09	11.76 ± 0.79	0.68 ± 0.11	26.31 ± 3.36
	NN	0.34 ± 0.09	0.70 ± 0.11	17.15 ± 1.90	0.71 ± 0.17	18.90 ± 2.06
	QDA	0.32 ± 0.05	0.17 ± 0.05	4.75 ± 1.26	0.62 ± 0.07	5.87 ± 1.36

Table 6. Runtimes in seconds for constructing a confidence set with ACORE for the Poisson example (top) and GMM example (bottom). The procedure for constructing confidence sets is outlined in Algorithm 4, and is split in 4 steps (see text). The rightmost column shows total runtimes.

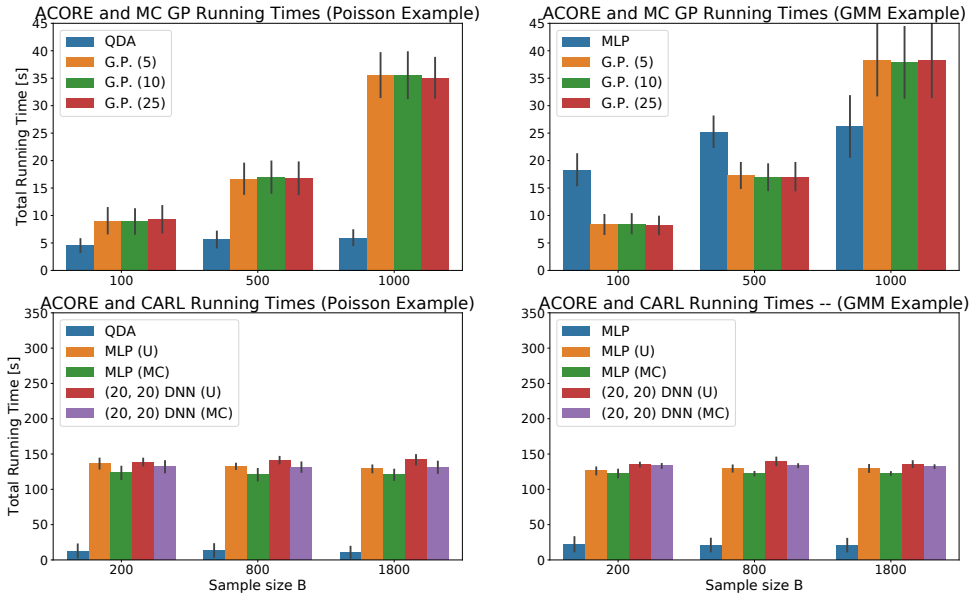

Figure 8. Runtimes in seconds for constructing a confidence set for the Poisson example (left panels) and GMM example (right panels). The best ACORE classifier runtime is compared with Gaussian process interpolation (GP) for $q = \{5, 10, 25\}$, and the two smaller CARL classifiers for both sampling schemes. See text for details. Confidence bars are built with a one standard deviation interval around the mean.