A calibration test for evaluating set-based epistemic uncertainty representations

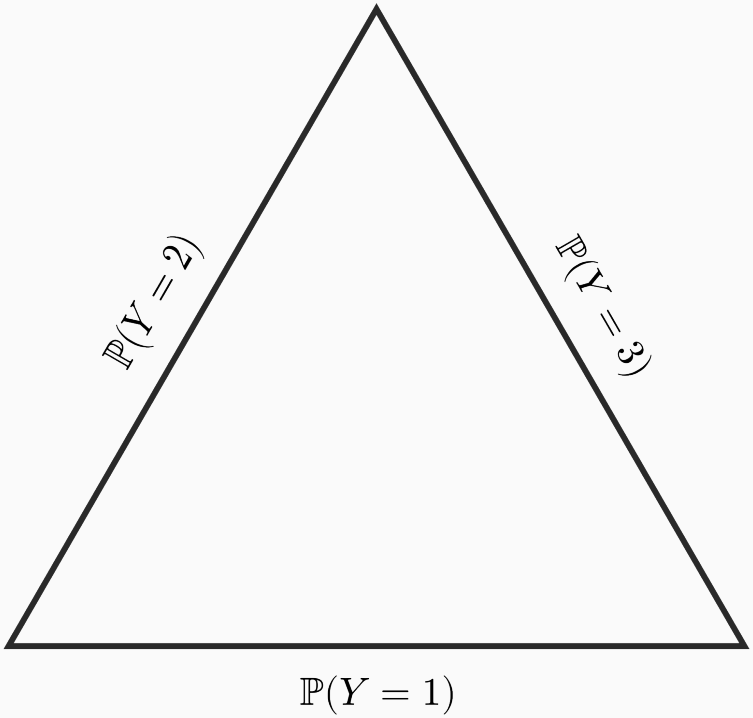
ECML PKDD 2025

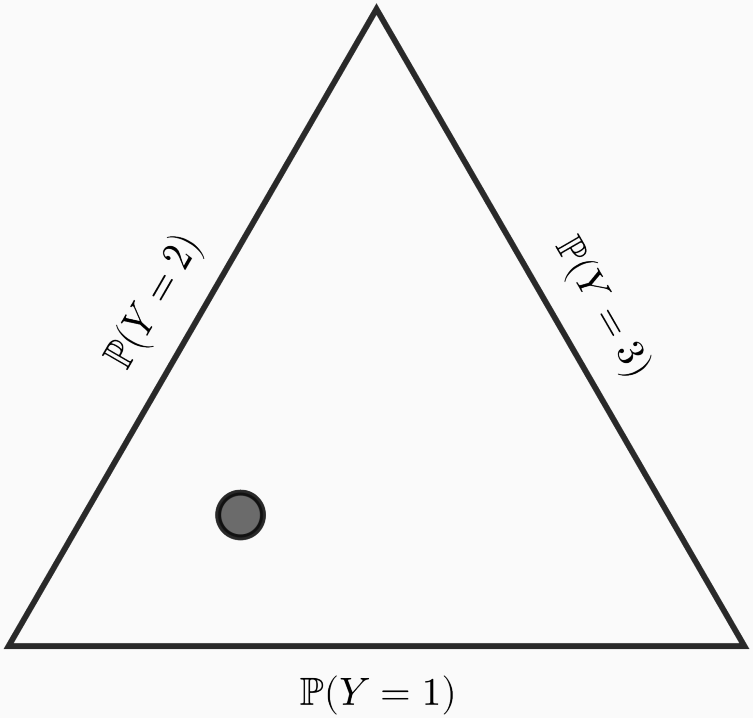
Mira Jürgens[1], Thomas Mortier[1], Viktor Bengs [2] Eyke Hüllermeier[2] and Willem Waegeman[1]

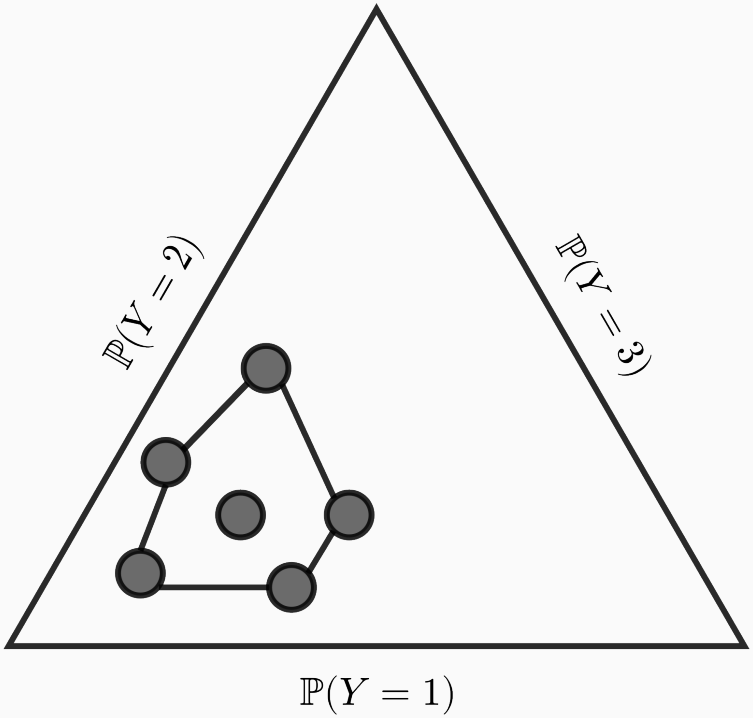
[1]Gent University [2]LMU Munich

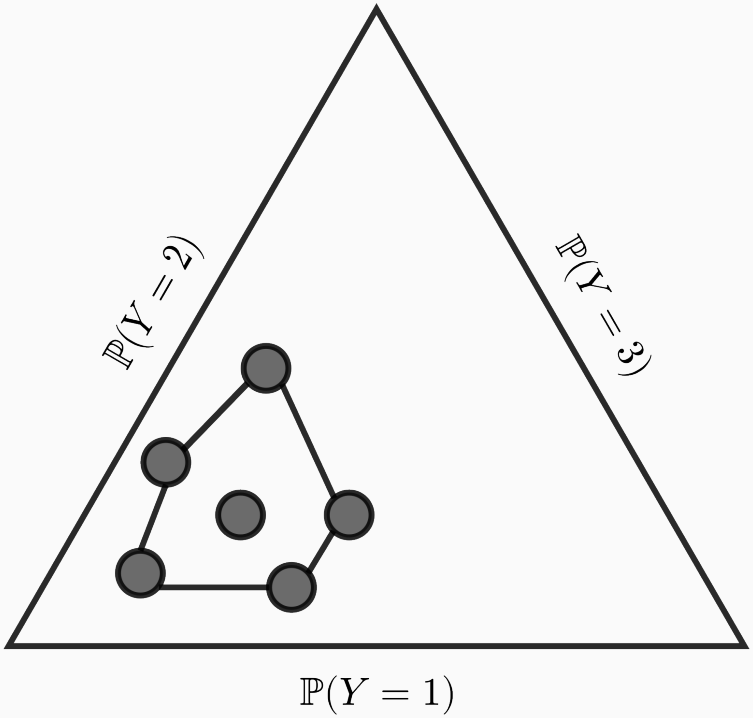
BioML

GHENT UNIVERSITY


$\mathbb{P}(Y=2)$
$\mathbb{P}(Y=3)$
$\mathbb{P}(Y=1)$


$\mathbb{P}(Y=2)$
$\mathbb{P}(Y=3)$
$\mathbb{P}(Y=1)$


$\mathbb{P}(Y=2)$
$\mathbb{P}(Y=3)$
$\mathbb{P}(Y=1)$



How do we know that set-based epistemic uncertainty representations "do what they should"?

Aleatoric and epistemic uncertainty in machine learning

Aleatoric vs epistemic uncertainty

Assume we have a probabilistic model $f : \mathcal{X} \to \mathbb{P}(\mathcal{Y})$ that estimates the conditional distribution $\mathbb{P}_{Y|X}$ of a r.v. Y given X. We can distinguish between **two types of uncertainty**:

Aleatoric vs epistemic uncertainty

Assume we have a probabilistic model $f : \mathcal{X} \to \mathbb{P}(\mathcal{Y})$ that estimates the conditional distribution $\mathbb{P}_{Y|X}$ of a r.v. Y given X. We can distinguish between **two types of uncertainty**:

Aleatoric uncertainty
irreducible randomness in $\mathbb{P}_{Y|X}$

Aleatoric vs epistemic uncertainty

Assume we have a probabilistic model $f : \mathcal{X} \to \mathbb{P}(\mathcal{Y})$ that estimates the conditional distribution $\mathbb{P}_{Y|X}$ of a r.v. Y given X. We can distinguish between **two types of uncertainty**:

Aleatoric uncertainty
irreducible randomness in $\mathbb{P}_{Y|X}$

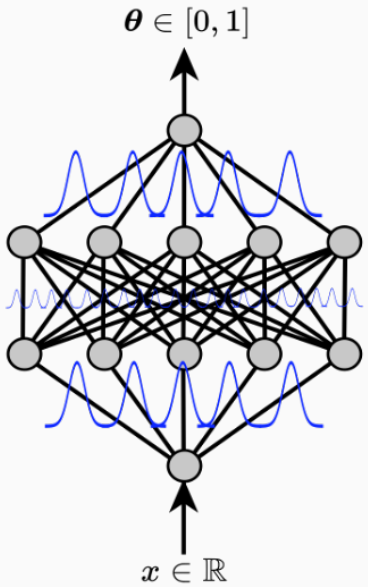
Epistemic uncertainty
reducible ignorance about the true $\mathbb{P}_{Y|X}$

Different representations of epistemic uncertainty

Distribution-based

second-order distribution $Q_{\theta|X}$

- e.g. Bayesian neural networks, evidential models, ensembles (approx.)

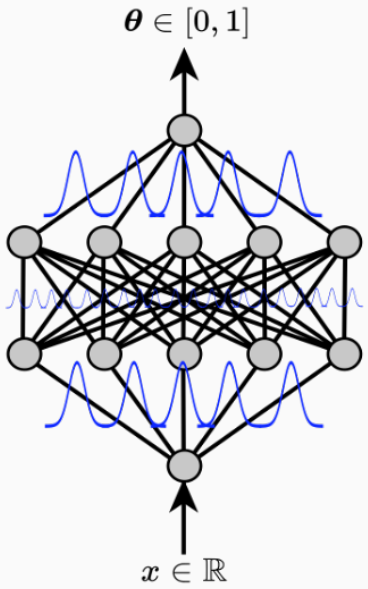


Different representations of epistemic uncertainty

Distribution-based

second-order distribution $Q_{\theta|X}$

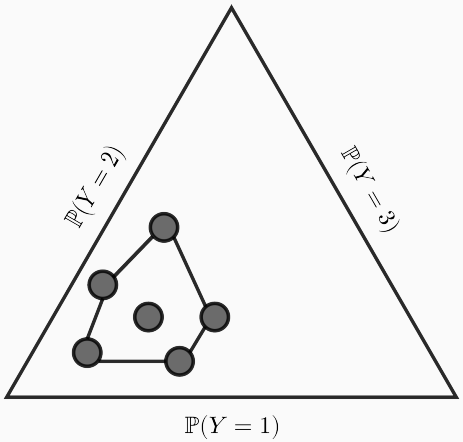
- e.g. Bayesian neural networks, evidential models, ensembles (approx.)



Set-based (focus today)

(convex) set $\mathcal{S}$ of probability distributions

- e.g. ensemble models, credal/interval networks



Validity of set-based uncertainty representations

Setting

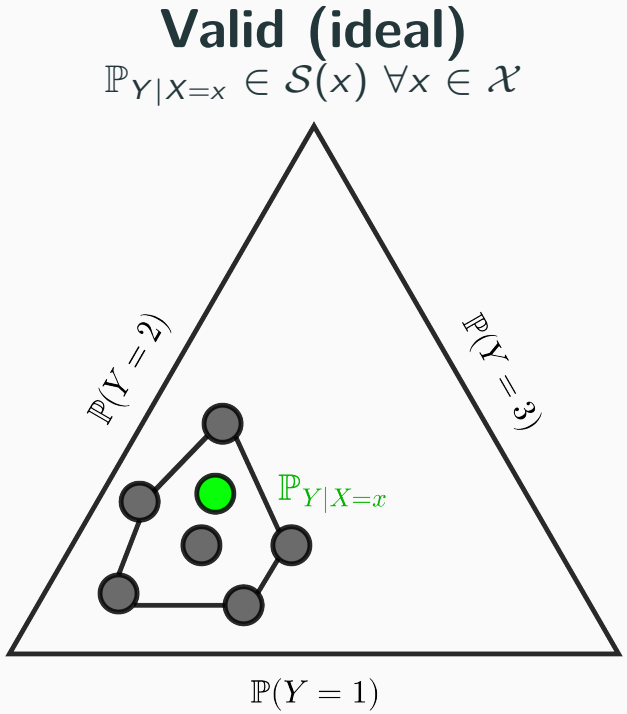
Notation	Meaning
$\mathcal{X} = \mathbb{R}^D$	instance space
$\mathcal{Y} = \{1, \ldots, K\}$	label space in multi-class classification
$\mathbb{P}_{X,Y}, \mathbb{P}_{Y\|X}$	joint/cond. prob. distribution of (X, Y), $Y\|X$ in $\mathcal{X} \times \mathcal{Y}$
$\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^N$	dataset of realisations
Δ_K	probability simplex for K classes
$f : \mathcal{X} \to \Delta_K$	probabilistic classifier model
$\mathcal{F} = \{f^{(1)}, \ldots, f^{(M)}\}$	ensemble of M different models
$\mathcal{S}(x)$	set of probability distributions on $\mathcal{Y}$ for a given x

When is a set-based representation valid?

Assume having a *ground truth* probability distribution $\mathbb{P}_{Y|X}$. Then for each $x \in \mathcal{X}$, we would like $\mathbb{P}_{Y|X=x}$ to be contained in the set $\mathcal{S}(x)$.

When is a set-based representation valid?

Assume having a *ground truth* probability distribution $\mathbb{P}_{Y|X}$. Then for each $x \in \mathcal{X}$, we would like $\mathbb{P}_{Y|X=x}$ to be contained in the set $\mathcal{S}(x)$.

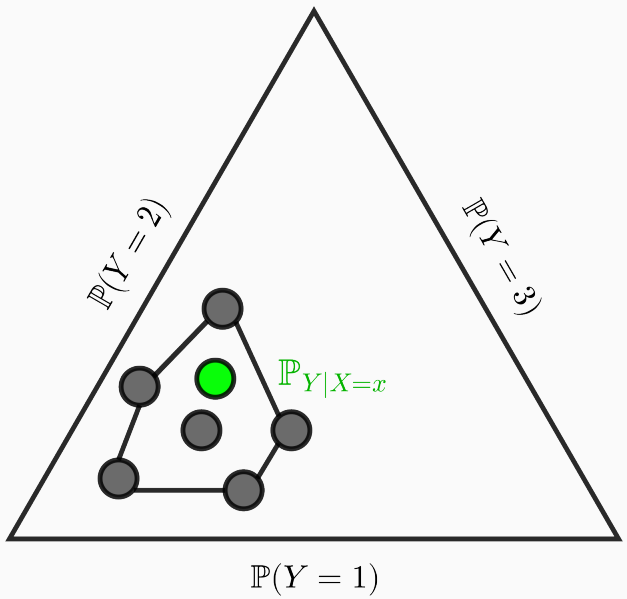


When is a set-based representation valid?

Assume having a *ground truth* probability distribution $\mathbb{P}_{Y|X}$. Then for each $x \in \mathcal{X}$, we would like $\mathbb{P}_{Y|X=x}$ to be contained in the set $\mathcal{S}(x)$.

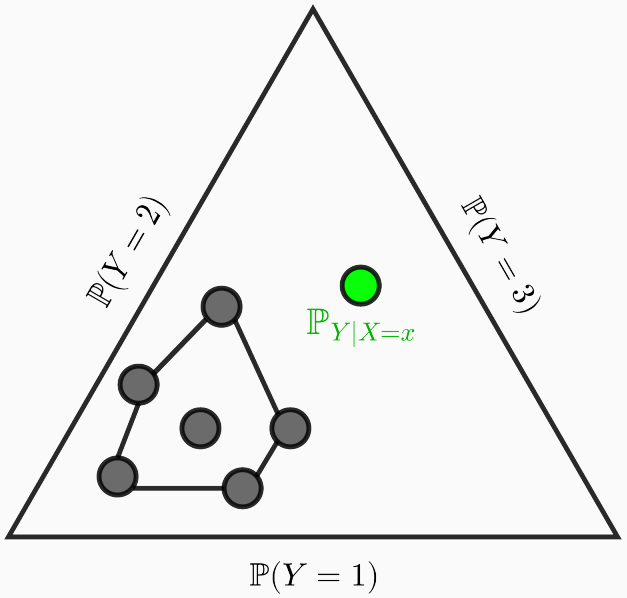
Valid (ideal)

$\mathbb{P}_{Y|X=x} \in \mathcal{S}(x)\ \forall x \in \mathcal{X}$



Invalid

$\exists x\, \mathbb{P}_{Y|X=x} \notin \mathcal{S}(x)$



How do we define $\mathcal{S}(x)$?

How do we define $\mathcal{S}(x)$?

Credal set formed by ensemble models

For $x \in \mathcal{X}$, let

$$\mathcal{F}|_x = \{f^{(1)}(x), \ldots, f^{(M)}(x)\} \subseteq \Delta_K$$

be a set of M different probability distributions on $\mathcal{Y}$. Define

$$\mathcal{S}(\mathcal{F}, x) = \{f_{\boldsymbol{\lambda}}(x) \in \Delta_K \big| f_{\boldsymbol{\lambda}}(x) = \sum_{i=1}^{M} \lambda_i(x) f^{(i)}(x)\}$$

as the set of all **convex combinations**, where $\boldsymbol{\lambda} : \mathcal{X} \to \Delta_M$ is a vector-valued function **from the instance space** to the M-simplex.

How do we define $\mathcal{S}(x)$?

Credal set formed by ensemble models

For $x \in \mathcal{X}$, let

$$\mathcal{F}|_x = \{f^{(1)}(x), \ldots, f^{(M)}(x)\} \subseteq \Delta_K$$

be a set of M different probability distributions on $\mathcal{Y}$. Define

$$\mathcal{S}(\mathcal{F}, x) = \{f_{\boldsymbol{\lambda}}(x) \in \Delta_K \big| f_{\boldsymbol{\lambda}}(x) = \sum_{i=1}^{M} \lambda_i(x) f^{(i)}(x)\}$$

as the set of all **convex combinations**, where $\boldsymbol{\lambda} : \mathcal{X} \to \Delta_M$ is a vector-valued function **from the instance space** to the M-simplex.

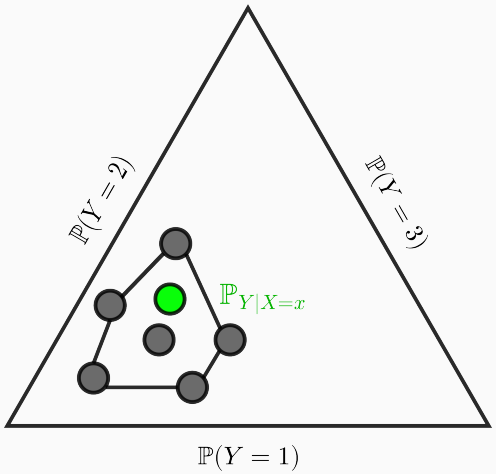
We say that $\mathcal{F}$ is *valid* iff

$$\mathbb{P}_{Y|X=x} \in \mathcal{S}(\mathcal{F}, x) \quad \text{for almost all } x \in \mathcal{X}.$$

How can we test that in practice?

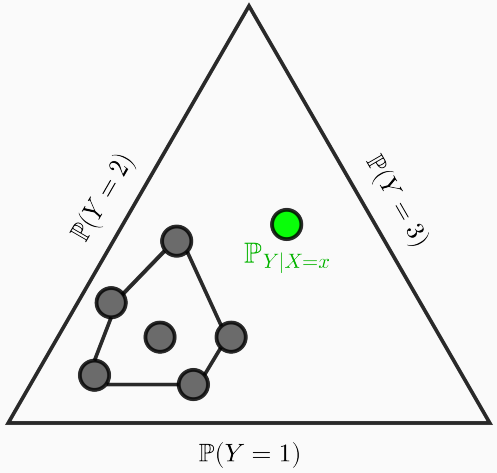
Valid (ideal)

$\mathbb{P}_{Y|X=x} \in \mathcal{S}$

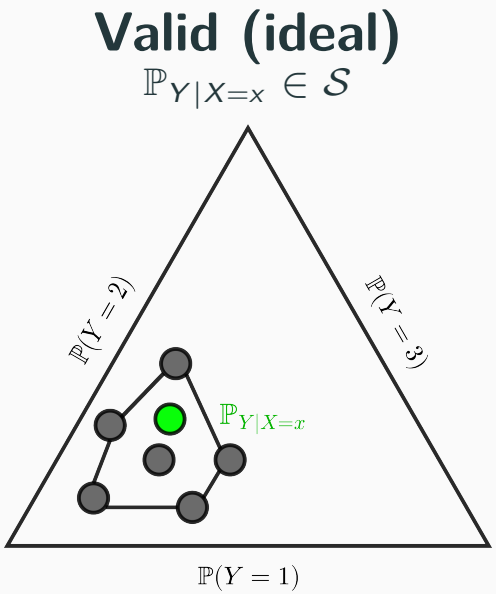


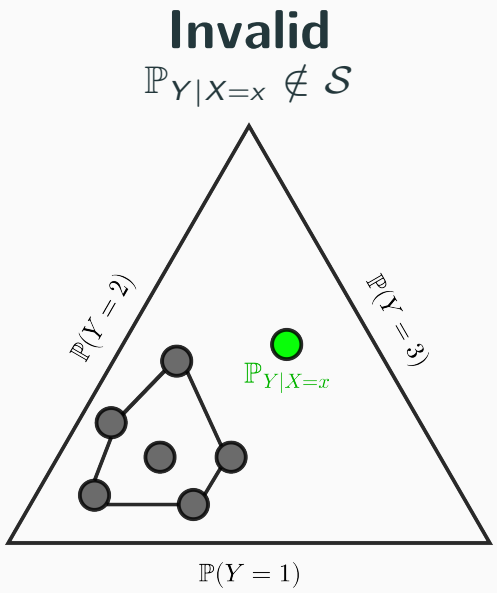
Invalid

$\mathbb{P}_{Y|X=x} \notin \mathcal{S}$



How can we test that in practice?



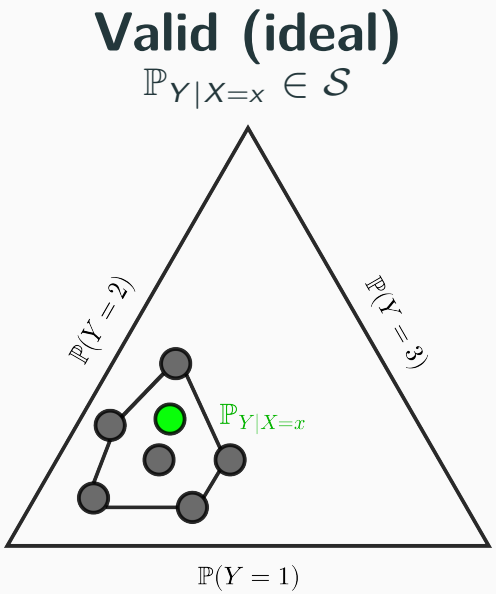


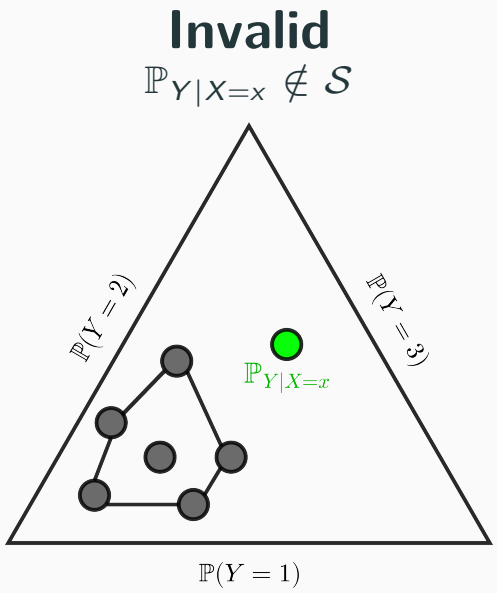
Typically not possible!

In classical settings, we **cannot observe or sample from** $\mathbb{P}_{Y|X=x}$.

We therefore test the ***necessary* condition**:

How can we test that in practice?





Typically not possible!

In classical settings, we **cannot observe or sample from** $\mathbb{P}_{Y|X=x}$.

We therefore test the ***necessary* condition**:

H_0 : $\forall x \, \exists \, \lambda(x) \in \Delta_M$ s.t. f_λ **is calibrated** vs H_1 : $\nexists \, \lambda(x)$.

Calibration as a distribution matching problem

Distribution calibration

A probabilistic classifier $f : \mathcal{X} \to \Delta^K$ is calibrated (in distribution) iff

$$\mathbb{E}\big[\mathbf{1}\{Y = k\} \,\big|\, f(X)\big] \;=\; f_k(X) \quad \text{a.s. for all } k.$$

Calibration as a distribution matching problem

Distribution calibration

A probabilistic classifier $f : \mathcal{X} \to \Delta^K$ is calibrated (in distribution) iff

$$\mathbb{E}\big[\mathbf{1}\{Y = k\} \,\big|\, f(X)\big] \;=\; f_k(X) \quad \text{a.s. for all } k.$$

Intuition: In *any* small **neighborhood** of score vectors $f(X)$, the observed class frequencies match the average predicted scores.

Calibration as a distribution matching problem

Distribution calibration

A probabilistic classifier $f : \mathcal{X} \to \Delta^K$ is calibrated (in distribution) iff

$$\mathbb{E}\big[\mathbf{1}\{Y = k\} \,\big|\, f(X)\big] \;=\; f_k(X) \quad \text{a.s. for all } k.$$

Intuition: In *any* small **neighborhood** of score vectors $f(X)$, the observed class frequencies match the average predicted scores.

How to "simulate" these local neighborhoods:

- **Binning.** Divide the score space into bins of equal width or mass.
  Disadvantages:
  - dependent on hard binning scheme
  - non-differentiable
  - only tests confidence and classwise, not distribution calibration

Calibration as a distribution matching problem

Distribution calibration

A probabilistic classifier $f : \mathcal{X} \to \Delta^K$ is calibrated (in distribution) iff

$$\mathbb{E}\big[\mathbf{1}\{Y = k\} \,\big|\, f(X)\big] \;=\; f_k(X) \quad \text{a.s. for all } k.$$

Intuition: In *any* small **neighborhood** of score vectors $f(X)$, the observed class frequencies match the average predicted scores.

How to "simulate" these local neighborhoods:

- **Binning.** Divide the score space into bins of equal width or mass.
  Disadvantages:
  - dependent on hard binning scheme
  - non-differentiable
  - only tests confidence and classwise, not distribution calibration
- **Our focus: kernel based estimators.** Use a kernel function to define a soft neighborhood.

Calibration error estimators use different distance measures

[1]Popordanoska et al., "A consistent and differentiable lp canonical calibration error estimator".

[2]Popordanoska et al., "Consistent and asymptotically unbiased estimation of proper calibration errors".

[3]Widmann et al., "Calibration tests in multi-class classification: A unifying framework".

[4]Kumar et al., "Trainable calibration measures for neural networks from kernel mean embeddings".

Calibration error estimators use different distance measures

1. **Expectation-over-divergence (EoD):**

$$\mathrm{CE}(f) = \mathbb{E}\Big[d\big(\mathbb{E}[\mathbf{1}\{Y = k\} \mid f(X)],\, f(X)\big)\Big].$$

   - $d = \|\cdot\|_{L_p}{}^1 \Rightarrow \mathrm{CE}_2$ for $p = 2$

[1] Popordanoska et al., "A consistent and differentiable lp canonical calibration error estimator".

[2] Popordanoska et al., "Consistent and asymptotically unbiased estimation of proper calibration errors".

[3] Widmann et al., "Calibration tests in multi-class classification: A unifying framework".

[4] Kumar et al., "Trainable calibration measures for neural networks from kernel mean embeddings".

Calibration error estimators use different distance measures

1. **Expectation-over-divergence (EoD):**

$$\mathrm{CE}(f) = \mathbb{E}\Big[d\big(\mathbb{E}[\mathbf{1}\{Y = k\} \mid f(X)],\, f(X)\big)\Big].$$

   - $d = \|\cdot\|_{L_p}{}^1 \Rightarrow \mathrm{CE}_2$ for $p = 2$
   - $d = d_{\mathrm{KL}}{}^2 \Rightarrow \mathrm{CE}_{\mathrm{KL}}$.

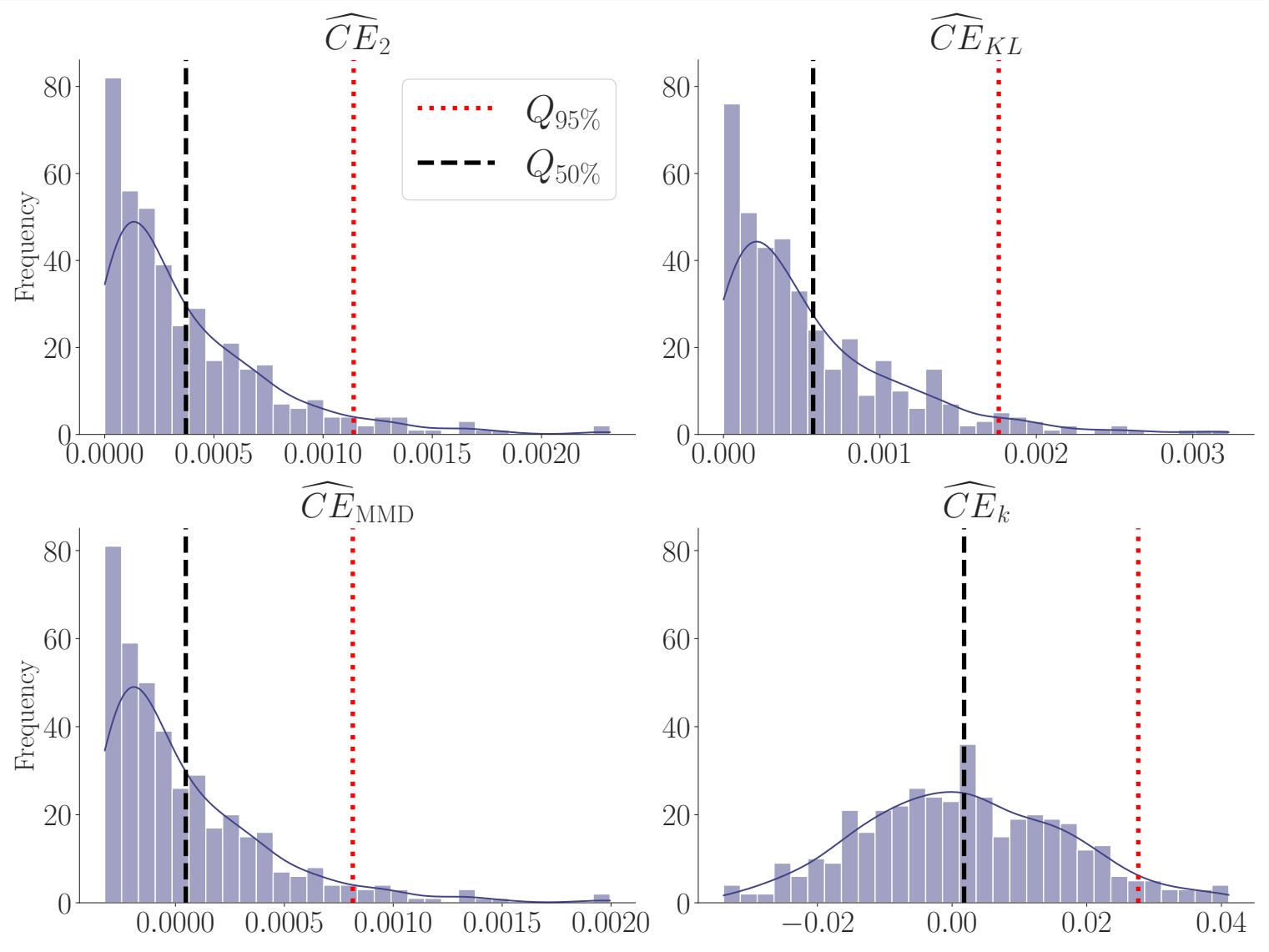
2. **Integral probability metric (IPM)**:

$$\mathrm{CE}(f) \quad = \quad \sup_{\phi \in \mathcal{H}} \Big|\mathbb{E}\big[\phi(f(X), Y)\big] - \mathbb{E}\big[\phi(f(X), Z)\big]\Big|,$$

   with $Z|f(X) \sim \mathrm{Cat}(f(X))$. Depending on the choicce of the kernel, we distinguish between CE_{k}[3] and $\mathrm{CE}_{\mathrm{MMD}}$[4].

[1] Popordanoska et al., "A consistent and differentiable lp canonical calibration error estimator".
[2] Popordanoska et al., "Consistent and asymptotically unbiased estimation of proper calibration errors".
[3] Widmann et al., "Calibration tests in multi-class classification: A unifying framework".
[4] Kumar et al., "Trainable calibration measures for neural networks from kernel mean embeddings".

Calibration estimators all follow different distributions under H_0



Calibration - set based perspective

Recall **the set of all instance-dependent convex combinations**:

$$\mathcal{S}(\mathcal{F}, x) = \{f_{\boldsymbol{\lambda}}(x) \in \mathcal{H} \Big| f_{\boldsymbol{\lambda}}(x). = \sum_{i=1}^{M} \lambda_i(x) f^{(i)}(x)\}$$

We defined our test setting as follows:

$H_0 : \ \forall x \, \exists \, \lambda(x) \in \Delta_M$ s.t. f_λ is calibrated vs $H_1 : \ \nexists \, \lambda(x).$

Calibration - set based perspective

Recall **the set of all instance-dependent convex combinations**:

$$\mathcal{S}(\mathcal{F}, x) = \{f_{\boldsymbol{\lambda}}(x) \in \mathcal{H} \big| f_{\boldsymbol{\lambda}}(x). = \sum_{i=1}^{M} \lambda_i(x) f^{(i)}(x)\}$$

We defined our test setting as follows:

$H_0 : \ \forall x \, \exists \, \lambda(x) \in \Delta_M$ s.t. f_λ is calibrated vs $H_1 : \ \nexists \, \lambda(x).$

Optimisation problem

Define the minimum calibration error over all convex combinations:

$$\min_{\boldsymbol{\lambda}: \mathcal{X} \to \Delta_M} g(f_{\boldsymbol{\lambda}}).$$

with $g \in \{\mathrm{CE}_2, \mathrm{CE}_{KL}, \mathrm{CE}_k, \mathrm{CE}_{MMD}\}$.

Optimisation - neural network based approach

Direct optimisation of calibration estimators is infeasible!

Optimisation - neural network based approach

Direct optimisation of calibration estimators is infeasible!

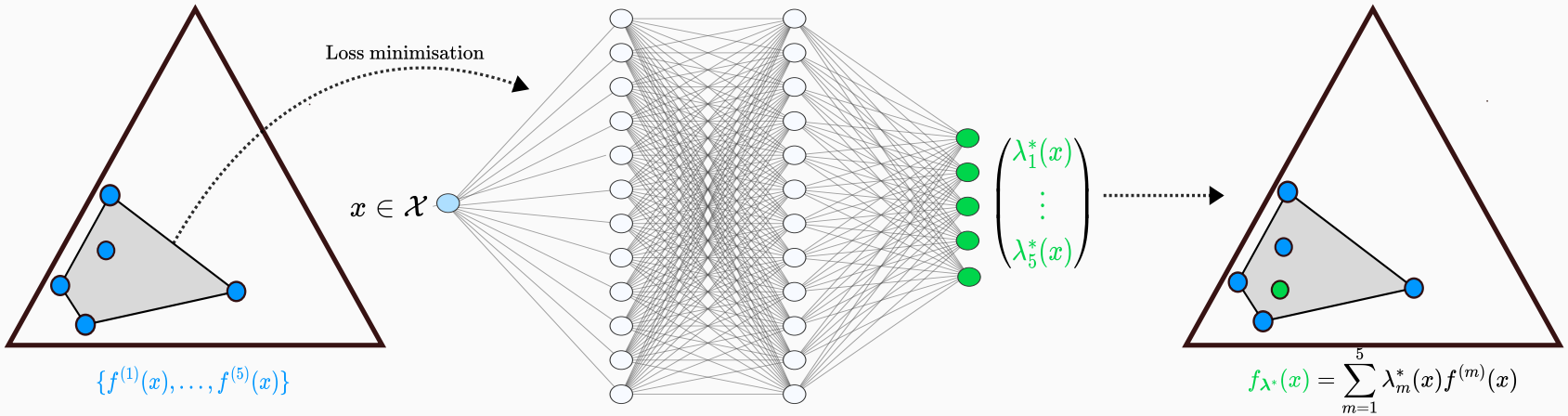
- calibration estimators are typically non-convex
- unidentifiability issue: many calibrated classifiers exist, $f(x) \equiv \mathbb{P}(Y)$ as a trivial solution

Optimisation - neural network based approach

Direct optimisation of calibration estimators is infeasible!

- calibration estimators are typically non-convex
- unidentifiability issue: many calibrated classifiers exist, $f(x) \equiv \mathbb{P}(Y)$ as a trivial solution

We propose a neural network based approach to learn the optimal convex combination.

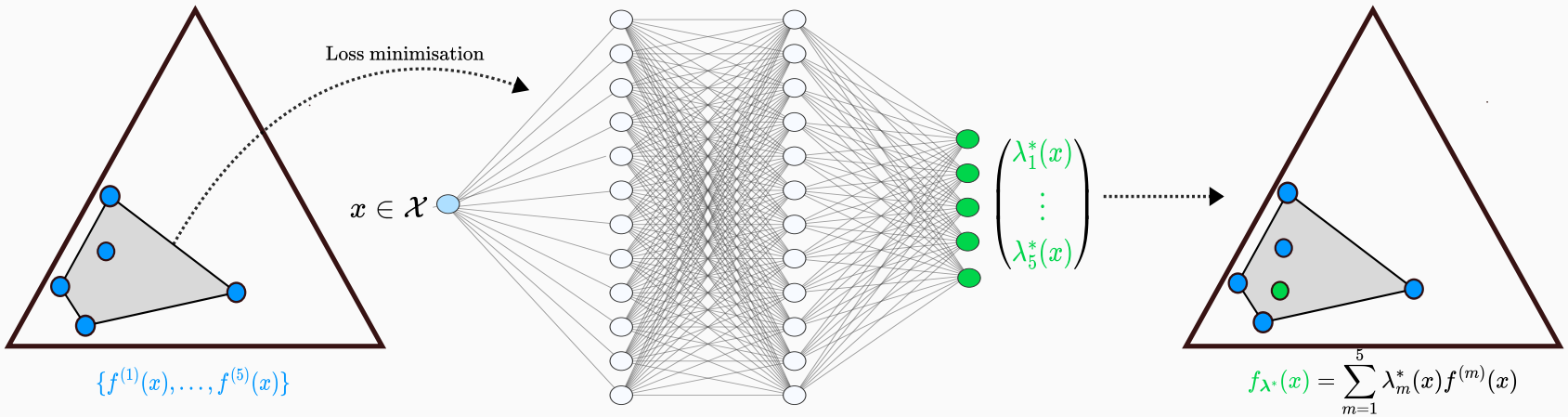


Optimisation - neural network based approach

Direct optimisation of calibration estimators is infeasible!

- calibration estimators are typically non-convex
- unidentifiability issue: many calibrated classifiers exist, $f(x) \equiv \mathbb{P}(Y)$ as a trivial solution

We propose a neural network based approach to learn the optimal convex combination.



Loss is a combination of a proper scoring rule ℓ with a calibration error term g:

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda}:\mathcal{X}\to\Delta_M}{\arg\min}\, \hat{\mathbb{E}}[\ell(Y, f_{\boldsymbol{\lambda}}(X))] + \beta \cdot g(f_{\boldsymbol{\lambda}}).$$

Algorithm 1 - an optimisation based approach to test calibration

1: **Input:** $\mathcal{D}_{val} = \{(x_i, y_i)\}_{i=1}^{N}$, $\mathcal{F}|_{(x_i)} = \left(f^{(1)}(x_i), \ldots, f^{(M)}(x_i)\right)$, $\hat{g}$, α, D

Algorithm 1 - an optimisation based approach to test calibration

1: **Input:** $\mathcal{D}_{val} = \{(x_i, y_i)\}_{i=1}^{N}$, $\mathcal{F}|_{(x_i)} = \left(f^{(1)}(x_i), \ldots, f^{(M)}(x_i)\right)$, $\hat{g}$, α, D

Optimization — learn $\lambda(x)$

1: $\Lambda^* \leftarrow \arg\min_{\Lambda} \hat{g}(f_\Lambda, \mathcal{D}_{val})$

2: $f_{\Lambda^*} \leftarrow \Lambda^* \cdot \mathcal{F}$

Algorithm 1 - an optimisation based approach to test calibration

Input: $\mathcal{D}_{val} = \{(x_i, y_i)\}_{i=1}^{N}$, $\mathcal{F}|_{(x_i)} = \left(f^{(1)}(x_i), \ldots, f^{(M)}(x_i)\right)$, $\hat{g}$, α, D

Optimization — learn $\lambda(x)$

$\Lambda^* \leftarrow \arg\min_{\Lambda} \hat{g}(f_\Lambda, \mathcal{D}_{val})$

$f_{\Lambda^*} \leftarrow \Lambda^* \overset{\Lambda}{\cdot} \mathcal{F}$

Bootstrap test — simulate calibrated labels (Vaicenavicius et al.)

for $d = 1, \ldots, D$ **do**

  $\mathcal{D}_d \leftarrow \{\tilde{x}_1, \ldots, \tilde{x}_N\}$ ▷ bootstrap instances

  for $n = 1, \ldots, N$ **do**

    sample $\tilde{y}_n \sim \mathrm{Cat}(f_{\Lambda^*}(x_n))$

  $\tilde{\mathcal{D}}_{val} \leftarrow \{(x_i, \tilde{y}_i)\}_{i=1}^{N}$ ▷ replace labels of validation set

  $t_{0,d} \leftarrow \hat{g}(f_{\Lambda^*}, \tilde{\mathcal{D}}_{val})$

Algorithm 1 - an optimisation based approach to test calibration

Input: $\mathcal{D}_{val} = \{(x_i, y_i)\}_{i=1}^{N}$, $\mathcal{F}|_{(x_i)} = \left(f^{(1)}(x_i), \ldots, f^{(M)}(x_i)\right)$, $\hat{g}$, α, D

Optimization — learn $\lambda(x)$

$\Lambda^* \leftarrow \arg\min_{\Lambda} \hat{g}(f_\Lambda, \mathcal{D}_{val})$

$f_{\Lambda^*} \leftarrow \Lambda^* \overset{\Lambda}{\cdot} \mathcal{F}$

Bootstrap test — simulate calibrated labels (Vaicenavicius et al.)

for $d = 1, \ldots, D$ **do**

  $\mathcal{D}_d \leftarrow \{\tilde{x}_1, \ldots, \tilde{x}_N\}$ ▷ bootstrap instances

  for $n = 1, \ldots, N$ **do**

    sample $\tilde{y}_n \sim \mathrm{Cat}(f_{\Lambda^*}(x_n))$

  $\tilde{\mathcal{D}}_{val} \leftarrow \{(x_i, \tilde{y}_i)\}_{i=1}^{N}$ ▷ replace labels of validation set

  $t_{0,d} \leftarrow \hat{g}(f_{\Lambda^*}, \tilde{\mathcal{D}}_{val})$

$q_{1-\alpha} \leftarrow (1-\alpha)$-quantile of $\{t_{0,d}\}_{d=1}^{D}$

$t \leftarrow \hat{g}(f_{\Lambda^*}, \mathcal{D}_{val})$

If $t > q_{1-\alpha}$ **then reject** H_0 **else do not reject**

Experimental results

Experimental setup - synthetic data

H_0 **true:**

$\boldsymbol{H_{0,1}}$**:** calibrated f_{λ^*} is a **constant** convex combination in $\mathcal{S}$

$\boldsymbol{H_{0,2}}$**:** f_{λ^*} is an **instance-dependent** combination in $\mathcal{S}$

Experimental setup - synthetic data

H_0 **true:**

- $\boldsymbol{H_{0,1}}$: calibrated f_{λ^*} is a **constant** convex combination in $\mathcal{S}$
- $\boldsymbol{H_{0,2}}$: f_{λ^*} is an **instance-dependent** combination in $\mathcal{S}$

H_1 **true:**

- $\boldsymbol{H_{1,1}}$: very close to the boundary of the spanned polytope ($\gamma = 0.05$)
- $\boldsymbol{H_{1,2}}$: at increased distance to the boundary ($\gamma = 0.2$)
- $\boldsymbol{H_{1,3}}$: at largest distance to the boundary ($\gamma = 0.8$)

Experimental setup - synthetic data

H_0 **true:**

$H_{0,1}$: calibrated f_{λ^*} is a **constant** convex combination in $\mathcal{S}$

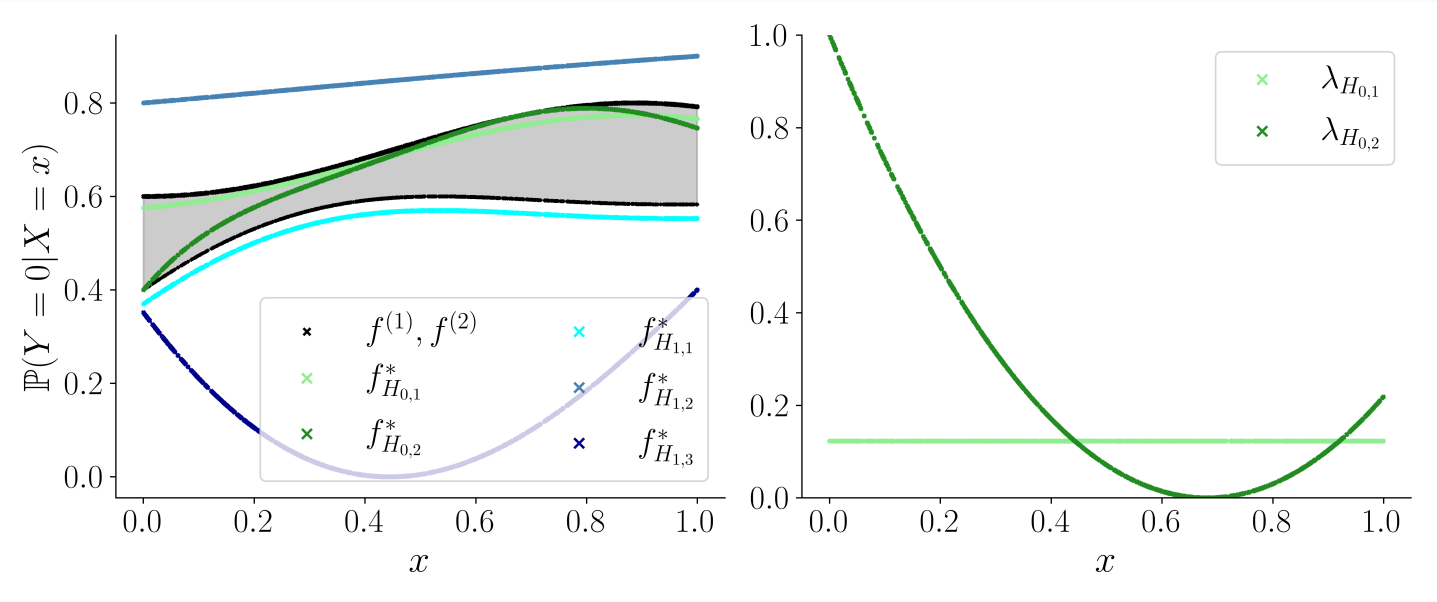
$H_{0,2}$: f_{λ^*} is an **instance-dependent** combination in $\mathcal{S}$

H_1 **true:**

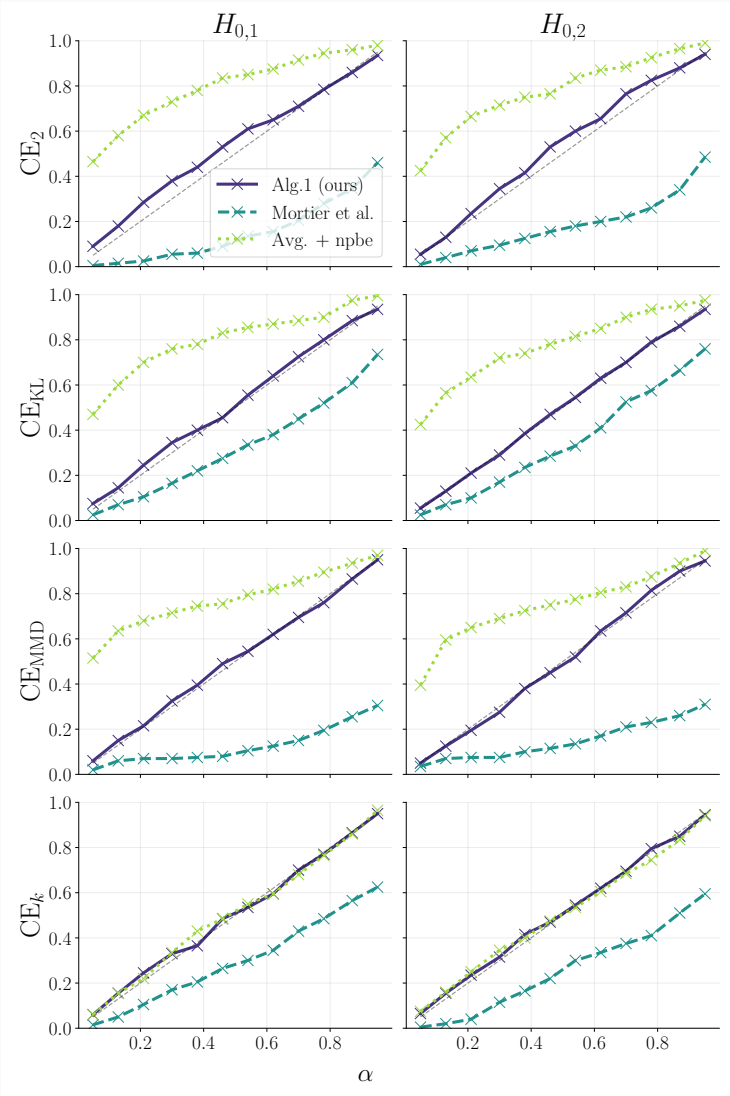
$H_{1,1}$: very close to the boundary of the spanned polytope ($\gamma = 0.05$)

$H_{1,2}$: at increased distance to the boundary ($\gamma = 0.2$)

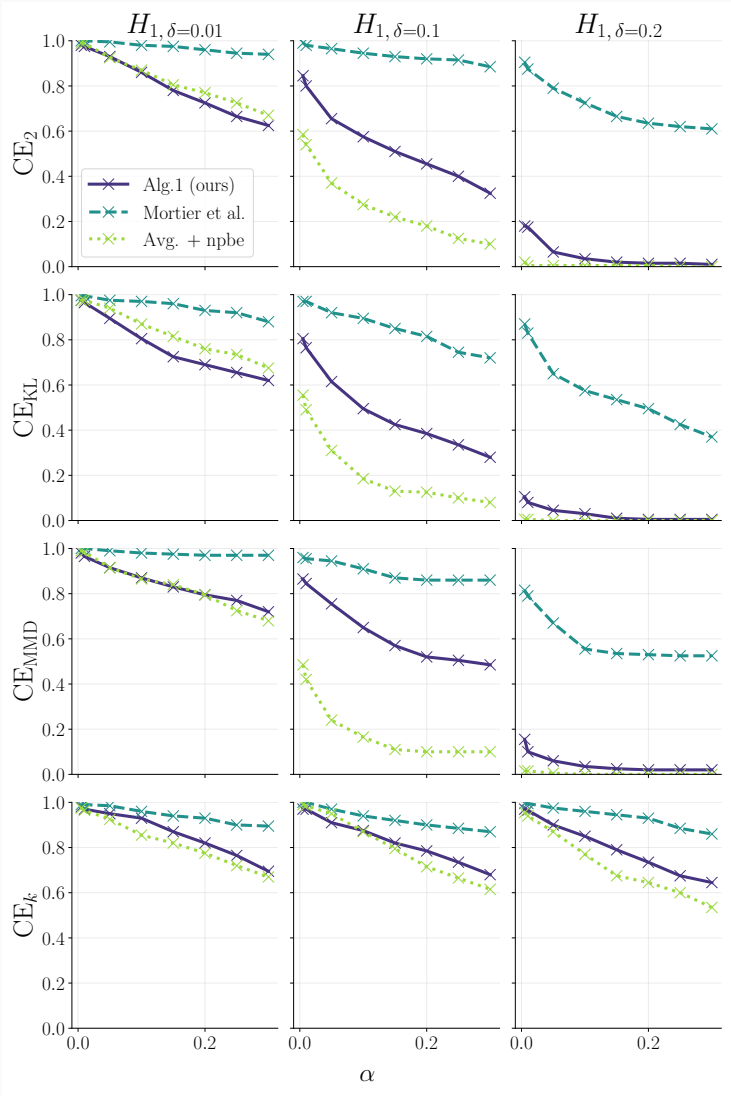
$H_{1,3}$: at largest distance to the boundary ($\gamma = 0.8$)



Exemplary experimental results - binary classification



(a) average Type 1 error



(b) average Type 2 error (Power)

Showcase on real data - CIFAR-10/100

Tests on Deep Ensemble, MC Dropout for VGG-16, ResNet-18 arhitectures.
In general:

- results on ensemble models (DE, MC Dropout) show very close proximity to the calibration test applied to mean estimate
- deep ensembles are more often calibrated than MC Dropout
- less complex model (ResNet-18) tends to be more often calibrated than more complex model (VGG-16).

$\Rightarrow$ **The test is able to flag several credal sets as invalid.**

Showcase on real data — CIFAR-10/100

Models: Deep Ensembles (DE), MC Dropout **Backbones:** VGG-16, ResNet-18

Key observations

- Calibration of DE / MC Dropout closely tracks calibration of the *mean* estimate.

Showcase on real data — CIFAR-10/100

Models: Deep Ensembles (DE), MC Dropout **Backbones:** VGG-16, ResNet-18

Key observations

- Calibration of DE / MC Dropout closely tracks calibration of the *mean* estimate.
- Deep Ensembles are calibrated more often than MC Dropout.

Showcase on real data — CIFAR-10/100

Models: Deep Ensembles (DE), MC Dropout **Backbones:** VGG-16, ResNet-18

Key observations

- Calibration of DE / MC Dropout closely tracks calibration of the *mean* estimate.
- Deep Ensembles are calibrated more often than MC Dropout.
- ResNet-18 is calibrated more often than VGG-16.

Showcase on real data — CIFAR-10/100

Models: Deep Ensembles (DE), MC Dropout **Backbones:** VGG-16, ResNet-18

Key observations

- Calibration of DE / MC Dropout closely tracks calibration of the *mean* estimate.
- Deep Ensembles are calibrated more often than MC Dropout.
- ResNet-18 is calibrated more often than VGG-16.

Takeaway

$\Rightarrow$ The test flags several *credal sets* as invalid.

$\Rightarrow$ More analysis on the impact of architecture, set based method, dataset is needed.

Conclusion and outlook

Conclusion

What we tested. Credal validity via a *necessary* criterion: does the credal set contain an *instance-dependent* calibrated convex combination?

Conclusion

What we tested. Credal validity via a *necessary* criterion: does the credal set contain an *instance-dependent* calibrated convex combination?

How. Learn $\lambda : \mathcal{X} \to \Delta_M$ by minimizing a *differentiable* calibration error; run bootstrap test on found optimal combination.

Conclusion

What we tested. Credal validity via a *necessary* criterion: does the credal set contain an *instance-dependent* calibrated convex combination?

How. Learn $\lambda : \mathcal{X} \to \Delta_M$ by minimizing a *differentiable* calibration error; run bootstrap test on found optimal combination.

What we see.

- On synthetic data: Type-1 $\approx \alpha$ and *higher power* than weight-sampling baselines. Instance-dependence becomes more relevant when credal sets are large.
- On real data: exemplary results on ensemble models (Deep Ensembles, MC Dropout) show applicability of the test; several credal sets are flagged as invalid.

Limitations. Calibration is *necessary, not sufficient*; finite-sample power depends on CE/kernel choices and the capacity of $\lambda(x)$; classification setting only.

Future prospects

GitHub

Paper

Interesting directions

- Application of the test to more diverse ensembles, credal set representations
- Extension to regression settings
- Usage of first order data representations
- Usage of local data neighborhoods to estimate *local* calibration