QUANTILE OPTIMIZATION FOR HONEST CONFORMAL PREDICTION UNDER LABEL SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Weighted conformal prediction (WCP) provides finite-sample coverage guarantees under label shift given true importance weights. However, under-coverage happens in practice when the discrepancy between estimated weights and the truth is not negligible, invalidating these guarantees. To restore validity, honest conformal prediction (HCP) evaluates the worst-case calibration sample quantile over a weight confidence region, yet this typical formulation poses a computational challenge, as it naturally requires non-convex fractional programming over n calibration samples. In this work, we present QuiCK (Quantile Computation for K-class HCP), an advanced computational suite that fundamentally fuses exact statistical honesty with practical computational efficiency. By decoupling the calibration sample size n from the class dimension K when evaluating empirical cumulative distribution functions, QuiCK leverages coordinate monotonicity for fast sequential optimization of quantile thresholds over marginal hyperboxes, and reformulates the original fractional programming form into a pure linear program (LP) over joint polytopes. Strategically, we streamline the optimization route for both cases by first computing a class-shared base solution, then efficiently localizing class-specific thresholds either by continuing the class-wise search directly from the base solution, or by applying LP sensitivity analysis to prune unaffected classes beforehand. Extensive experiments on synthetic datasets and real-world benchmarks validate that QuiCK strictly achieves honest conformal prediction guarantees. With a `C++` and `OpenMP` backend, QuiCK delivers exact statistical honesty while maintaining computational efficiency.

1 INTRODUCTION

Conformal prediction has been established as a distribution-free uncertainty quantification framework in modern machine learning, constructing prediction sets with rigorous *finite-sample* coverage guarantees (Vovk et al., 2005; Lei et al., 2018; Angelopoulos & Bates, 2023). However, its underlying assumption, data exchangeability, can be sabotaged by label shift issues in practical deployments, where the class proportions vary between source and target environments (Lipton et al., 2018; Azizzadenesheli et al., 2019; Garg et al., 2020; Alexandari et al., 2020; Saerens et al., 2002; Tian et al., 2023). To adapt, Weighted Conformal Prediction (WCP) reweights calibration score samples using importance weights, which are ratios that reflect the class proportion changes between source and target domains (Podkopaev & Ramdas, 2021). While WCP achieves exact $1-\alpha$ coverage when importance weights are perfectly known, this oracle assumption rarely holds in practice. Because, in reality, weights must be estimated from finite labeled source data and unlabeled target data, it inevitably introduces statistical estimation errors. Such errors break finite-sample coverage guarantees, and it has been theoretically quantified in counterfactual and robust conformal inference settings (Lei & Candès, 2021; Jin et al., 2023).

If distribution-free coverage guarantees are to be preserved under label shift, the uncertainty quantification for prediction sets must remain strictly honest, leaving no room for coverage gaps, a principle referred to as honest conformal prediction (HCP). While Li et al. (2026) account for weight estimation errors in finite-sample setting by evaluating the worst-case calibration sample quantile over a valid confidence region of importance weights, approximating this quantile via Monte Carlo sampling fails to preserve the exact *finite-sample* coverage guarantees. Optimizing this worst-case quantile can be naturally cast as a fractional programming (FP) task, but the heavy computational

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

burden is real. Traditionally, FP problems can be addressed by Dinkelbach's Algorithm (Dinkelbach, 1967), or through the Charnes-Cooper Transformation to reformulate the FP into a linear program (LP) (Charnes & Cooper, 1962; Schaible, 1995; Stancu-Minasian, 2012). Most importantly, searching for the quantile requires *repeatedly* evaluating the empirical cumulative distribution function (eCDF) under varying candidate weights. At the sample level, every change in the weights requires a full $O(n)$ scan of the n calibration samples, even when only one single class weight changes.

To uncover the real underlying structure of the quantile optimization task, we separate the static sample prefix sums from the dynamic K-class weight coefficients. Armed with this fast eCDF evaluation tool, we strategically streamline the quantile optimization for all classes as a whole, presenting QuiCK (Quantile Computation for K-class HCP), an advanced computational suite. QuiCK can handle two *finite-sample* confidence regions. First, when class weights are modeled independently without cross-class dependence, the confidence region forms an axis-aligned hyperbox, known as marginal confidence region. QuiCK leverages this marginal geometry and quantile monotonicity, yielding a fast sequential optimization procedure for all HCP quantile thresholds together. Second, when accounting for complex inter-class dependencies, the confidence region generalizes to a polytope, known as Joint confidence region (Li et al., 2026). QuiCK uses a pure LP formulation, which bypasses FP entirely and leads to a coherent development for computing HCP quantile thresholds for all classes simultaneously.

In summary, we challenge the status quo of computational efficiency in honest conformal prediction. We believe that finite-sample guarantees can be restored under label shift in an efficient manner, free from oracle assumptions and robust to plug-in errors. By investigating the fundamental connection between weighted sample quantiles and corresponding eCDFs, we uncover the true mathematical structure of the worst-case quantile optimization, and decoupling dynamic class weights from static calibration score samples. Starting from a fast coordinate-wise solver on hyperboxes to exact pure LP reformulations over polytopes, we materialize these theoretical insights into QuiCK, achieving high computational efficiency by computing class-specific threshold simultaneously.

2 Problem formulation & honest conformal prediction

2.1 Notations

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the feature space and $\mathcal{Y} = \{1, \ldots, K\} \equiv [K]$ be the label space. We consider a source population $\mathcal{P}$ and a target population $\mathcal{Q}$ on $\mathcal{X} \times \mathcal{Y}$. Write the corresponding probability density/mass functions in $\mathcal{P}$ and $\mathcal{Q}$ as $p(\cdot)$ and $q(\cdot)$ respectively. For $y \in \mathcal{Y}$, let $p_{\mathbf{X}|Y}(\mathbf{x}, y)$ and $p_Y(y)$ denote the class-conditional probability density/mass function (pdf/pmf) of $\mathbf{X}$ given $Y = y$ and the marginal pmf of Y in $\mathcal{P}$, respectively, and define $q_{\mathbf{X}|Y}(\mathbf{x}, y)$ and $q_Y(y)$ analogously under $\mathcal{Q}$. Let $\mathrm{pr}(\cdot)$ denote the probability of an event, with the underlying distribution indicated by a subscript when needed, for example $\mathrm{pr}_{(\mathbf{X},Y)\sim\mathcal{P}}(\cdot)$ and $\mathrm{pr}_{(\mathbf{X},Y)\sim\mathcal{Q}}(\cdot)$. In the unsupervised label shift setting, we observe a labeled source dataset $\mathcal{D}_{\mathrm{src}} = \{(\mathbf{X}_i, Y_i)\}_{i=1}^{n_s}$ drawn i.i.d. from $\mathcal{P}$ and an unlabeled target dataset $\mathcal{D}_{\mathrm{tgt}} = \{\mathbf{X}_i\}_{i=n_s+1}^{n_s+n_t}$ drawn i.i.d. from the target population $\mathcal{Q}$, and we use $\mathcal{Q}_{\mathbf{X}}$ to denote the marginal distribution of $\mathbf{X}$ in $\mathcal{Q}$ and n_s and n_t are sample sizes. The label shift assumption states that the class-conditional distributions are invariant across the source and target domains, namely, $p_{\mathbf{X}|Y}(\mathbf{x} \mid y) = q_{\mathbf{X}|Y}(\mathbf{x} \mid y)$ for $y \in \mathcal{Y}$, while the marginal label distributions may differ, that is, $p_Y(y) \neq q_Y(y)$. Under this assumption, the importance weights $\boldsymbol{\omega}^* = (\omega_1^*, \ldots, \omega_K^*)$ are defined by $\omega_y^* = q_Y(y)/p_Y(y)$, $y \in \mathcal{Y}$, and $\boldsymbol{\omega}^* \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$, where $\mathbb{R}_+ \equiv [0, \infty)$.

2.2 Split conformal prediction under no distribution shift

Conformal prediction aims to find a prediction set $\mathcal{C}(\mathbf{X}_0) \subseteq \mathcal{Y}$ of Y_0 for a new observation $(\mathbf{X}_0, Y_0)$ using the observed data. In practice, split conformal prediction was introduced to address the high computational cost of the original "full" conformal framework (Vovk et al., 2005; Lei et al., 2015). Specifically, in a classification task ($\mathcal{Y} = [K]$) without distribution shifts, the split conformal prediction procedure finds a prediction set for a new observation $(\mathbf{X}_0, Y_0) \sim \mathcal{P}$ as following:

1. **Data Splitting:** The available dataset, $\mathcal{D}_{\mathrm{src}}$, is partitioned into a training set $\mathcal{D}_{\mathrm{train}}$ and a calibration set $\mathcal{D}_{\mathrm{cali}} = \{\mathbf{X}_i, Y_i\}_{i=1}^{n}$.

2. **Model Fitting:** Train a soft-max classifier $\mathbf{g} : \mathcal{X} \to [0,1]^K$ on $\mathcal{D}_{\text{train}}$ and construct a nonconformity score function $r(\cdot,\cdot) : \mathcal{X} \times \mathcal{Y} \to [0,\infty)$ using $\mathbf{g}$.
3. **Calibration:** Compute the nonconformity scores on $\mathcal{D}_{\text{cali}}$ and get $\{r(\mathbf{X}_i, Y_i)\}_{i=1}^n$.
4. **Finding Threshold:** Compute ζ, the $\lceil (1-\alpha)(n+1) \rceil$-th order statistic of $\{r(\mathbf{X}_i, Y_i)\}_{i=1}^n \cup \{\infty\}$.
5. **Constructing a Prediction Set:** The prediction set of Y_0 for a new test point $(\mathbf{X}_0, Y_0) \sim \mathcal{P}$ is then formed as $\{Y_0 : r(\mathbf{X}_0, Y_0) \leq \zeta\}$.

The nonconformity score function $r(\cdot,\cdot)$ in Model Fitting step is constructed subjectively. Since the soft-max classifier $\mathbf{g}(\mathbf{x}) = \{g_1(\mathbf{x}), ..., g_K(\mathbf{x})\}$ approximates $\{p_{Y|\mathbf{X}}(1,\mathbf{x}), ..., p_{Y|\mathbf{X}}(K,\mathbf{x})\}$, the common choice for the nonconformity score $r(\mathbf{x}, y)$ is $1 - g_y(\mathbf{x})$ (Sadinle et al., 2019) or $\sum_{k=1}^K g_k(\mathbf{x}) I\{g_k(\mathbf{x}) \geq g_y(\mathbf{x})\}$ (Romano et al., 2020). We use a randomized version $r(\mathbf{x}, y) = \sum_{k=1}^K g_k(\mathbf{x}) \big[I\{g_k(\mathbf{x}) > g_y(\mathbf{x})\} + U I\{g_k(\mathbf{x}) = g_y(\mathbf{x})\} \big]$ with $U \sim \text{Uniform}(0,1)$, which makes $r(\mathbf{x}, y)$ a continuous function and guarantees $r(\mathbf{X}, Y) \mid \mathbf{X} \approx \text{Uniform}(0,1)$ (Podkopaev & Ramdas, 2021).

In the above procedure, Finding Threshold step can be recast as follows: Let F be the CDF of the discrete probability measure $(n+1)^{-1}\{\sum_{i=1}^n \delta_{r(\mathbf{X}_i, Y_i)} + \delta_\infty\}$, where δ_c denotes the Dirac delta measure at $c \in \mathbb{R} \cup \{\infty\}$. Then $\zeta = Q_{1-\alpha}(F) = \inf\{t : F(t) \geq 1-\alpha\}$, where $Q_{1-\alpha}$ denotes the $(1-\alpha)$-quantile of F. This recasting simplifies the adjustment for $(\mathbf{X}_0, Y_0) \sim \mathcal{Q}$ when label shift is considered, which is described below.

2.3 Split conformal prediction under label shift

Standard conformal prediction relies on the assumption of exchangeability, which is violated in the presence of distribution shift. Tibshirani et al. (2019) address this introducing a weighted exchangeability framework for covariate shift, and Podkopaev & Ramdas (2021) propose weighted conformal prediction under label shift. To perform conformal prediction under label shift, we retain Data Splitting, Model Fitting, and Calibration steps from the above procedure, while replacing Finding Threshold with computing the $(1-\alpha)$-quantile of $F(\cdot; \boldsymbol{\omega}, y)$, where

$$F(\cdot; \boldsymbol{\omega}, y) = \sum_{i=1}^n \delta_{r(\mathbf{X}_i, Y_i)} \frac{\omega_{Y_i}}{\sum_{i=1}^n \omega_{Y_i} + \omega_y} + \delta_\infty \frac{\omega_y}{\sum_{i=1}^n \omega_{Y_i} + \omega_y}, \tag{1}$$

using importance weights, $\boldsymbol{\omega} = (\omega_1, ..., \omega_K)$, and a certain $y \in [K]$. Explicitly,

$$\zeta(y; \boldsymbol{\omega}) \equiv Q_{1-\alpha}\{F(\cdot; \boldsymbol{\omega}, y)\} = \inf\{t : F(t; \boldsymbol{\omega}, y) \geq 1-\alpha\}.$$

Consequentially, we replace Constructing a Prediction Set with forming $\mathcal{C}(\mathbf{X}_0; \boldsymbol{\omega}) \equiv \{Y_0 : r(\mathbf{X}_0, Y_0) \leq \zeta(Y_0; \boldsymbol{\omega})\}$ for a new test point $(\mathbf{X}_0, Y_0) \sim \mathcal{Q}$. Note that $\zeta(y; \omega)$ is a weighted sample quantile of $\{r(\mathbf{X}_i, Y_i)\}_{i=1}^n \cup \{\infty\}$, with the weight depending on the y under consideration.

When the true importance weights $\boldsymbol{\omega}^*$ are known, the prediction set constructed above has a valid prediction coverage, as established in Theorem 2 of Podkopaev & Ramdas (2021). We rewrite the result as $\text{pr}_{(\mathbf{X}_0, Y_0) \sim \mathcal{Q}}\{r(\mathbf{X}_0, Y_0) \leq \zeta(Y_0; \boldsymbol{\omega}^*) \mid \mathcal{D}_{\text{train}}\} \geq 1 - \alpha$.

Because the true importance weights $\boldsymbol{\omega}^*$ are usually unknown in practice, one usually relies on estimated importance weights $\widehat{\boldsymbol{\omega}} = (\widehat{\omega}_1, ..., \widehat{\omega}_K) \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$ derived from the observed data $\mathcal{D}_{\text{src}} \cup \mathcal{D}_{\text{tgt}}$. Then a similar prediction set $\{Y_0 : r(\mathbf{X}_0, Y_0) \leq \zeta(Y_0; \widehat{\boldsymbol{\omega}})\}$ can be constructed, referred to as plug-in WCP. However, a problem arises when we use $\widehat{\boldsymbol{\omega}}$ instead of $\boldsymbol{\omega}^*$: the randomness in $\widehat{\boldsymbol{\omega}}$ creates the *coverage gap* (the amount the true coverage is less than the desired $1-\alpha$) of the prediction set. The coverage gap depends on the estimation accuracy of $\widehat{\boldsymbol{\omega}}$ and thus is determined by the data quality, the sample sizes, and the estimation procedure.

Remark 1. *The notation $\boldsymbol{\omega}$ denotes any vector in $\mathbb{R}_+^K \setminus \{\mathbf{0}\}$ and does not necessarily correspond to valid importance weights. In particular, $\sum_{y=1}^K p(y)\omega_y$ need not be 1, whereas the true weight function $\boldsymbol{\omega}^*$ satisfies $\sum_{y=1}^K p_Y(y)\omega_y^* = \sum_{y=1}^K q_Y(y) = 1$.*

2.4 Honest conformal prediction (HCP) under label shift

Instead of using $\zeta(Y_0; \widehat{\boldsymbol{\omega}})$ in plug-in WCP, HCP considers the worst-case quantile over the confidence region of $\boldsymbol{\omega}^*$. Concretely, let $\boldsymbol{\Omega} \subseteq \mathbb{R}_+^K \setminus \{\mathbf{0}\}$ be a $(1-\epsilon)$-level confidence region for $\boldsymbol{\omega}^*$, *i.e.*,

$\text{pr}\{\boldsymbol{\omega}^* \in \boldsymbol{\Omega} \mid \mathcal{D}_{\text{train}}\} \geq 1-\epsilon$, then the worst-case quantile $\zeta(y;\boldsymbol{\Omega}) \equiv \sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$ is used as a threshold. The corresponding prediction set is $\mathcal{C}(\mathbf{x};\boldsymbol{\Omega}) \equiv \{y : r(\mathbf{x},y) \leq \zeta(y;\boldsymbol{\Omega})\}$, for $\mathbf{x} \in \mathcal{X}$. Equivalently, $\mathcal{C}(\mathbf{x};\boldsymbol{\Omega}) = \bigcup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \mathcal{C}(\mathbf{x};\boldsymbol{\omega})$, hence the prediction set is conservative compared to the prediction set at any single fixed $\boldsymbol{\omega} \in \boldsymbol{\Omega}$.

When we target $1-\alpha$ coverage prediction set, we adjust $(1-\alpha)$-quantile in Finding Threshold with the $(1-\alpha+\epsilon)$-quantile to account for the uncertainty in the importance weights through $\boldsymbol{\Omega}$. Formally, we establish the $1-\alpha$ coverage guarantee when $\zeta(Y_0;\boldsymbol{\Omega})$ is used as the threshold in Theorem 1.

Theorem 1 (Honest conformal prediction under label shift)**.** *Suppose that* $pr\{\boldsymbol{\omega}^* \in \boldsymbol{\Omega} \mid \mathcal{D}_{\text{train}}\} \geq 1-\epsilon$ *for* $0 < \epsilon < \alpha$*. Let* $\beta = \alpha - \epsilon$*. For* $y \in [K]$ *and* $\boldsymbol{\omega} \in \boldsymbol{\Omega}$*, define* $\zeta(y;\boldsymbol{\omega}) = Q_{1-\beta}F(\cdot;\boldsymbol{\omega},y)$ *and* $\zeta(y;\boldsymbol{\Omega}) = \sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$*. Then* $pr_{(\mathbf{X}_0,Y_0)\sim\mathcal{Q}}\{r(\mathbf{X}_0,Y_0) \leq \zeta(Y_0;\boldsymbol{\Omega}) \mid \mathcal{D}_{\text{train}}\} \geq 1-\alpha$*.*

The result follows because the event $\boldsymbol{\omega}^* \notin \boldsymbol{\Omega}$ occurs with probability at most ϵ, and the event $r(\mathbf{X}_0,Y_0) > \zeta(Y_0;\boldsymbol{\omega}^*)$ occurs with probability at most $\beta = \alpha - \epsilon$. We provide the proof of Theorem 1 in Appendix J.1, and the methods to construct $\boldsymbol{\Omega}$ using $\mathcal{D}_{\text{src}}$ and $\mathcal{D}_{\text{tgt}}$ are provided in Li et al. (2026) with the summary in Appendix F.

3 Computation of $\zeta(y;\boldsymbol{\Omega})$

A natural question is how we can compute the threshold $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$ over $\boldsymbol{\Omega}$ for each potential $y \in [K]$, based on $\mathcal{D}_{\text{cali}}$ and the additional covariate $\mathbf{x} \in \mathcal{X}$. Here, we aim to compute the *exact* worst-case sample quantile, $\zeta(y;\boldsymbol{\Omega})$, over $\boldsymbol{\Omega}$ instead of approximating it by Monte Carlo sampling.

Let $\boldsymbol{\omega}^{\max}(y) \equiv \arg\max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$ denote the importance weights leading to the maximum threshold $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$. For preparation, for each $k \in [K]$, define $N_k(r) = \sum_{i=1}^n I\{Y_i = k, r(\mathbf{X}_i,Y_i) \leq r\}$ for $r \in \mathbb{R}$. Let $n_k \equiv \sum_{i=1}^n I\{Y_i = k\}$ be the number of observations in class k. For a fixed y, define $n_k(y) \equiv n_k + I\{y = k\}$, the number of observations in class k after including the candidate label y. Then $\sum_{k=1}^K n_k = n$ and $\sum_{k=1}^K n_k(y) = n+1$.

Define the CDFs $F_k(\cdot)$ and $F_k(\cdot;y)$ associated with class k as $F_k(r) = N_k(r)/n_k, F_k(r;y) = N_k(r)/n_k(y)$ for $r \in \mathbb{R}$. When $n_k = 0$ or $n_k(y) = 0$, define $F_k(r) = 0$ and $F_k(r;y) = 0$, respectively, for all r. Then by definition, for $k \neq y$, $F_k(r) = F_k(r;y)$ and for $k = y$, $F_y(r;y) = F_y(r)n_y/(n_y+1)$. Define $\tau_k \equiv Q_{1-\beta}\{F_k(\cdot)\}$ and $\tau_k(y) \equiv Q_{1-\beta}\{F_k(\cdot;y)\}$. Note that τ_k and $\tau_k(y)$ are the $(1-\beta)$-quantiles of the observations with $Y = k$, $k \in [K]$. Here, $F_k(\cdot)$, $F_k(\cdot;y)$, τ_k, and $\tau_k(y)$ do not depend on $\boldsymbol{\omega}$.

Recall $F(\cdot;\boldsymbol{\omega},y)$ is defined as $F(r;\boldsymbol{\omega},y) = \sum_{i=1}^n I\{r(\mathbf{X}_i,Y_i) \leq r\}\omega_{Y_i}/(\sum_{i=1}^n \omega_{Y_i} + \omega_y)$, and $\zeta(y;\boldsymbol{\omega}) = Q_{1-\beta}\{F(\cdot;\boldsymbol{\omega},y)\}$. Subsequently, $F(\cdot;\boldsymbol{\omega},y)$ can be written in terms of $N_k(\cdot)$ as $F(r;\boldsymbol{\omega},y) = \sum_{k=1}^K N_k(r)\omega_k/\sum_{k=1}^K n_k(y)\omega_k$. We further define the CDF without the new observation $(\mathbf{x},y)$ as $F(\cdot;\boldsymbol{\omega})$, where

$$F(r;\boldsymbol{\omega}) \equiv \frac{\sum_{i=1}^n I\{r(\mathbf{X}_i,Y_i) \leq r\}\omega_{Y_i}}{\sum_{i=1}^n \omega_{Y_i}} = \frac{\sum_{k=1}^K N_k(r)\omega_k}{\sum_{k=1}^K n_k\omega_k}. \tag{2}$$

We define $\zeta(\boldsymbol{\omega}) \equiv Q_{1-\beta}\{F(\cdot;\boldsymbol{\omega})\}$ and $\boldsymbol{\omega}^{\max} \equiv \arg\max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(\boldsymbol{\omega})$. Throughout, quantities defined after incorporating the candidate observation $(\mathbf{x},y)$ include y as an argument, whereas their counterparts based only on the calibration observations do not.

Before detailing QuiCK, Figure 4 in Appendix I illustrates how varying sample weights impact the weighted sample quantile, under a simplified setting of one sample per class. Also, we provide the geometric characterization of $\boldsymbol{\omega}^{\max}(y)$ in Appendix D.

3.1 QuiCK.hyperbox: marginal confidence region

3.1.1 Base algorithm for hyperbox

We first consider when $\boldsymbol{\Omega}$ is a hyperbox in $\mathbb{R}_+^K \setminus \{\mathbf{0}\}$, *i.e.*, $\boldsymbol{\Omega} = \prod_{k=1}^K [\underline{\omega}_k, \overline{\omega}_k]$. The base problem uses only the calibration observations, without augmenting the sample with the candidate observation $(\mathbf{x},y)$ or imposing $r(\mathbf{x},y) = \infty$. To search for the largest $\zeta(\boldsymbol{\omega})$, using (2), the intuition is that

we should increase the effect of those $F_k(\cdot)$'s that have large quantile τ_k values. Thus, when $\mathbf{\Omega}$ is hyperbox, it is sensible to assign $\overline{\omega}_k$ to these k's, while assigning $\underline{\omega}_k$ to the rest. Let $\sigma : [K] \to [K]$ be a permutation such that $\tau_{\sigma(1)} \leq \cdots \leq \tau_{\sigma(K)}$. We need to identify a separation point c such that $\omega_{\sigma(k)} = \underline{\omega}_{\sigma(k)}$ for $k \leq c$ and $\omega_{\sigma(k)} = \overline{\omega}_{\sigma(k)}$ for $k > c$.

We perform the search sequentially, starting from $\boldsymbol{\omega} = (\underline{\omega}_1, \ldots, \underline{\omega}_K)$. Then, sequentially for $\sigma(K), \sigma(K-1), \ldots$, we change $\underline{\omega}_{\sigma(k)}$ to $\overline{\omega}_{\sigma(k)}$ and verify whether the resulting $\zeta(\boldsymbol{\omega})$ value increases. We stop as soon as this value decreases, which occurs if the τ_k value whose weight is to be increased is smaller than the current quantile $\zeta(\boldsymbol{\omega})$. Algorithm QuiCK.hyperbox.base provides the detail of the optimization procedure.

In Algorithm QuiCK.hyperbox.base, given that $\zeta(\boldsymbol{\omega})$ is the most expensive computation in the entire procedure, the algorithm requires evaluating $\zeta(\boldsymbol{\omega}^{(t)}) \geq \tau_{\sigma(K-t)}$ instead of $\zeta(\boldsymbol{\omega}^{(t)}) \geq \zeta(\boldsymbol{\omega}^{(t+1)})$ as a stopping criterion, leveraging the precomputed τ_k values and avoids the potential computation of $\zeta(\boldsymbol{\omega}^{(t+1)})$ (see Appendix G for detailed discussion).

Theorem 2. *Let* $\mathbf{\Omega} = \prod_{k=1}^{K} [\underline{\omega}_k, \overline{\omega}_k]$. *Then Algorithm QuiCK.hyperbox.base gives* $\sup_{\boldsymbol{\omega} \in \mathbf{\Omega}} \zeta(\boldsymbol{\omega})$.

We summarize the proof steps in Figure 5 and provide the proof in Appendix J.2.

3.1.2 Simultaneous quantile optimization for all classes

We extend Algorithm QuiCK.hyperbox.base to consider all $y \in [K]$ simultaneously. Note that $\tau_k(y) = \tau_k$ for $k \neq y$ and $\tau_y(y) = Q_{1-\beta}\{F_y(\cdot; y)\} \geq \tau_y = Q_{1-\beta}\{F_y(\cdot)\}$. In fact, since

$$F_y(r; y) \geq 1 - \beta \iff N_y(r) \geq (n_y + 1)(1 - \beta) \iff F_y(r) \geq \frac{(n_y + 1)(1 - \beta)}{n_y},$$

$\tau_y(y)$ is a quantile of $F_y(\cdot)$ as $\tau_y(y) = Q_{(1-\beta)(n_y+1)/n_y}\{F_y(\cdot)\}$. Thus, for each candidate class y, only the threshold τ_y is replaced by the weakly larger threshold $\tau_y(y)$, while the remaining $K - 1$ thresholds are unchanged. Consequently, the relative ordering among all classes $k \neq y$ remains unchanged, and class y can only move weakly to the right in the ordering.

This limited change in the ordering allows us to use the base solution $\sup_{\boldsymbol{\omega} \in \mathbf{\Omega}} \zeta(\boldsymbol{\omega})$ as a warm start. Let t_0 denote the exit index obtained from Algorithm QuiCK.hyperbox.base, so that $\zeta(\boldsymbol{\omega}^{(t_0-1)}) < \tau_{\sigma(K-t_0+1)}$ and $\zeta(\boldsymbol{\omega}^{(t_0)}) \geq \tau_{\sigma(K-t_0)}$. Since $F(r; \boldsymbol{\omega}) \geq F(r; \boldsymbol{\omega}, y)$ for any $r \in \mathbb{R}$, $\boldsymbol{\omega} \in \mathbf{\Omega}$, and $y \in [K]$, it follows that $\zeta(\boldsymbol{\omega}) \leq \zeta(y; \boldsymbol{\omega})$.

The base solution is particularly informative when n_y is sufficiently large. Augmenting the sample changes the quantile level for class y by only $(1 - \beta)/n_y$, while $F(r; \boldsymbol{\omega}, y)$ approaches $F(r; \boldsymbol{\omega})$ as the calibration sample size n increases. Thus, when the weighted eCDF $F(\cdot; \boldsymbol{\omega})$ is sufficiently regular around the $(1 - \beta)$-quantile, $\tau_y(y)$ is close to τ_y and $\zeta(y; \boldsymbol{\omega})$ is close to $\zeta(\boldsymbol{\omega})$. In this case, the exit index for the augmented problem is expected to be near the base exit index t_0.

Theorem 3. *Let* $\mathbf{\Omega} = \prod_{k=1}^{K} [\underline{\omega}_k, \overline{\omega}_k]$. *Then Algorithm QuiCK.hyperbox gives* $\sup_{\boldsymbol{\omega} \in \mathbf{\Omega}} \zeta(y; \boldsymbol{\omega})$ *for all* $y \in [K]$.

3.2 QuiCK.polytope: joint confidence region

We next consider a convex polytope $\mathbf{\Omega}$, which is described by $\mathbf{\Omega} = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega} \leq \mathbf{b}, \boldsymbol{\omega} \geq \mathbf{0}\}$ as shown in Appendix F, where $\mathbf{A}$ is a $2K \times K$ matrix and $\mathbf{b}$ is a $2K$-dimensional vector, and $\leq$ denotes coordinate-wise inequalities.

3.2.1 Base algorithm for polytope

We also first consider the base problem without a new observation y. One approach to quantile optimization over a convex polytope is to directly minimize the weighted eCDF $F(r; \boldsymbol{\omega})$ at each candidate threshold $r = \zeta(\boldsymbol{\omega}^{(t)})$ using fractional programming, as in Algorithm 5 of Appendix H. Here, we instead consider the sorted calibration scores to find $\sup_{\boldsymbol{\omega} \in \mathbf{\Omega}} \zeta(\boldsymbol{\omega})$.

Let $r_{(1)} \leq r_{(2)} \leq \cdots \leq r_{(n)}$ denote the ordered values of $\{r(\mathbf{X}_1, Y_1), \ldots, r(\mathbf{X}_n, Y_n)\}$, and let $Y_{(j)}$ and $\omega_{(j)}$ be the Y_i and ω_{Y_i} corresponding to $r_{(j)} = r(\mathbf{X}_i, Y_i)$. Using this notation, we can write

$N_k(r_{(t)})$ as counts for each class k and the weighted eCDF $F(r_{(t)};\boldsymbol{\omega})$ as the fraction of prefix sums $\sum_{i=1}^{n} I\{r_{(i)} \le r_{(t)}\}\omega_{(i)}$ and $\sum_{i=1}^{n}\omega_{(i)}$, *i.e.*,

$$N_k(r_{(t)}) = \sum_{i=1}^{n} I\{Y_{(i)} = k, r_{(i)} \le r_{(t)}\}, \quad F(r_{(t)};\boldsymbol{\omega}) = \frac{\sum_{i=1}^{n} I\{r_{(i)} \le r_{(t)}\}\omega_{(i)}}{\sum_{i=1}^{n}\omega_{(i)}}.$$

Under this notation, we optimize the computation for the base problem from two aspects. The first is the transformation of the threshold check to a linear programming problem. Since the only possible values of $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega})$ are $r(\mathbf{X}_i, Y_i)$ for $i = 1,\dots,n$, we only need to determine whether $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega})$ exceeds $r_{(t)}$ for $t = 1, ..., n$. By definition of $(1-\beta)$-quantile, $\zeta(\boldsymbol{\omega}) > r_{(t)}$ is equivalent to the weighted eCDF at $r_{(t)}$ falling strictly below $1-\beta$, *i.e.*, $F(r_{(t)};\boldsymbol{\omega}) < 1-\beta$. Then $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega}) > r_{(t)}$ if and only if $\exists\,\boldsymbol{\omega}\in\boldsymbol{\Omega}$ s.t. $F(r_{(t)};\boldsymbol{\omega}) < 1-\beta$. By (2), this is further equivalent to $g_t(\boldsymbol{\omega}) > 0$ for some $\boldsymbol{\omega}\in\boldsymbol{\Omega}$, where $g_t(\boldsymbol{\omega}) \equiv \sum_{k=1}^{K}\left\{(1-\beta)n_k - N_k(r_{(t)})\right\}\omega_k$. Hence, for each $t\in[n]$, this condition can be checked by solving the linear program

$$\max_{\boldsymbol{\omega}} \sum_{k=1}^{K}\left\{(1-\beta)n_k - N_k(r_{(t)})\right\}\omega_k \quad \text{subject to} \quad \mathbf{A}\boldsymbol{\omega} \le \mathbf{b} \quad \boldsymbol{\omega} \ge \mathbf{0}.$$

The optimal objective value $f_t \equiv \max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} g_t(\boldsymbol{\omega})$ then determines whether $r_{(t)}$ exceeds $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega})$.

(i) If $f_t > 0$, then there exists a feasible weight vector $\boldsymbol{\omega}\in\boldsymbol{\Omega}$ such that $\zeta(\boldsymbol{\omega})$ exceeds $r_{(t)}$, *i.e.*, $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega}) > r_{(t)}$. Then we search $r_{(i)}$ for $i > t$.

(ii) If $f_t \le 0$, then for all feasible weight vectors $\boldsymbol{\omega}\in\boldsymbol{\Omega}$, $\zeta(\boldsymbol{\omega}) \le r_{(t)}$, and hence $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega}) \le r_{(t)}$. Then we search r_i for $i < t$.

Moreover, f_t is nonincreasing in $t\in[n]$. Indeed, $N_k(r_{(t)})$ is nondecreasing in t for every $k\in[K]$, so $g_t(\boldsymbol{\omega})$ is nonincreasing in t for every fixed $\boldsymbol{\omega}\ge\mathbf{0}$. Taking the maximum over $\boldsymbol{\omega}\in\boldsymbol{\Omega}$ preserves this ordering. Hence, there exists a smallest index l_0 such that $f_{l_0} \le 0$.

Second, using this monotonicity of f_t, we apply binary search to find the smallest $t\in[n]$ such that $f_t \le 0$. If $f_m > 0$ at the midpoint m, then the optimal threshold lies strictly above $r_{(m)}$, so we continue searching over indices larger than m. Otherwise, the optimal threshold is no larger than $r_{(m)}$, so we continue searching over indices no larger than m.

The optimal solution $r_{(l_0)}$ and reduced costs $\check{c}_k(l_0)$ are retained for the simultaneous optimization described in the next subsection. If $\boldsymbol{\lambda}\ge\mathbf{0}$ denotes the optimal dual variable associated with $\mathbf{A}\boldsymbol{\omega}\le\mathbf{b}$ when solving the linear program at $r_{(t)}$, the reduced cost of ω_k is $\check{c}_k(t) \equiv \{(1-\beta)n_k - N_k(r_{(t)})\} - \mathbf{A}_{\cdot k}^{\mathrm{T}}\boldsymbol{\lambda}$ for $k\in[K]$, where $\mathbf{A}_{\cdot k}$ denotes the k-th column of $\mathbf{A}$.

Theorem 4. *Let* $\boldsymbol{\Omega} = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega} \le \mathbf{b}, \boldsymbol{\omega} \ge \mathbf{0}\}$ *be a convex polytope. Then Algorithm QuiCK.polytope.base returns* $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega})$ *in at most* $\lceil log_2(n+1)\rceil$ *iterations.*

3.2.2 Simultaneous quantile optimization using LP sensitivity

In order to simultaneously obtain $l_0(y)$ for all $y\in[K]$, we first solve the base problem without y and apply linear programming sensitivity analysis to identify how the objective function f_t changes after incorporating y.

For a candidate threshold $r_{(t)}$, define the objective function $g_t(\boldsymbol{\omega}, y)$ as $g_t(\boldsymbol{\omega}, y) \equiv \sum_{k=1}^{K}\left\{(1-\beta)n_k(y) - N_k(r_{(t)})\right\}\omega_k$. Since $n_k(y) = n_k$ for $k \ne y$ and $n_y(y) = n_y + 1$, the augmented objective satisfies $g_t(\boldsymbol{\omega}, y) = g_t(\boldsymbol{\omega}) + (1-\beta)\omega_y$. Thus, relative to the base problem, incorporating the candidate class y amounts to increasing only the objective coefficient associated with ω_y by $1-\beta$.

Let $l_0(y)$ denote the smallest index t such that $\max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} g_t(\boldsymbol{\omega}, y) \le 0$. Then $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(y;\boldsymbol{\omega}) = r_{(l_0(y))}$. Since $F(r;\boldsymbol{\omega}, y) \le F(r;\boldsymbol{\omega})$ for every $r\in\mathbb{R}$ and $\boldsymbol{\omega}\in\boldsymbol{\Omega}$, we have $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(y;\boldsymbol{\omega}) \ge \sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(\boldsymbol{\omega})$, and hence $l_0(y) \ge l_0$. Therefore, once the base problem has been solved, the search for $l_0(y)$ only needs to consider indices $t \ge l_0$.

We further exploit the linear programming sensitivity analysis for the perturbation in the objective function coefficient. Let $\boldsymbol{\omega}^{(l_0)}$ be an optimal solution to the base linear program at $r_{(l_0)}$, and let $\check{c}_k(l_0)$ denote the corresponding reduced costs. If the y-th component $\omega_y^{(l_0)} = 0$ and $\check{c}_y(l_0) + (1 - \beta) \leq 0$, then $\check{c}_y(l_0) < 0$, so $\omega_y^{(l_0)}$ is nonbasic and increasing its objective coefficient by $1 - \beta$ does not change the optimal basis. In this case, $\omega_y^{(l_0)}$ remains zero, so the optimal objective value at $r_{(l_0)}$ is unchanged and $l_0(y) = l_0$. Hence, no additional linear program needs to be solved for this class y.

If this sensitivity condition is not satisfied, we search for the smallest $t \geq l_0$ such that $\max_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} g_t(\boldsymbol{\omega}, y) \leq 0$. Because this optimal objective value is nonincreasing in t, the search can proceed sequentially or, more efficiently, by binary search.

The effect of adding the candidate class y can also be characterized directly through the optimal value of the linear program. Since $f_t(y) \equiv \max_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} g_t(\boldsymbol{\omega}, y) = \max_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} \{g_t(\boldsymbol{\omega}) + (1-\beta)\omega_y\}$, we have $f_t \leq f_t(y) \leq f_t + (1-\beta) \sup_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} \omega_y$. Thus, incorporating a single candidate observation perturbs the optimal LP value by at most a bounded amount that does not accumulate with the calibration sample size n. In particular, the sign of f_t cannot change whenever $f_t \leq -(1-\beta) \sup_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} \omega_y$, so only indices for which the base LP objective function f_t lies sufficiently close to zero can affect the new exit index $l_0(y)$. When n is large and the f_t crosses zero sharply around $t \approx l_0$, then $l_0(y)$ is expected to remain close to l_0. Thus, the base solution l_0 provides a natural warm start when the sensitivity condition does not immediately certify $l_0(y) = l_0$. Algorithm QuiCK.polytope summarizes the simultaneous procedure.

Theorem 5. *Let* $\boldsymbol{\Omega} = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega} \leq \mathbf{b}, \boldsymbol{\omega} \geq \mathbf{0}\}$ *be a convex polytope. Then Algorithm QuiCK.polytope gives* $\sup_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} \zeta(y; \boldsymbol{\omega})$ *for all* $y \in [K]$.

4 Empirical Validation

We investigate the practical performance of QuiCK through diagnostic synthetic datasets and real-world benchmarks. Our empirical experiments are driven by two primary objectives: first, to verify the *statistical honesty* of our framework, demonstrating its ability to strictly guarantee nominal coverage under label shift; and second, to profile *computational efficiency* of our methods by comparing runtime to alternative strategies and analyzing its optimization complexity.

Throughout the experiments, we compare QuiCK against two fundamental baselines and a modified sampling method: (i) Oracle WCP: An idealized baseline that utilizes the true importance weights, serving as the gold standard for target coverage; (ii) Plug-in WCP: A standard baseline that directly applies point-estimated weights on the quantile computation, a practice potentially yielding undercoverage (Podkopaev & Ramdas, 2021); (iii) Boundary-Biased Hit-and-Run (BBHR): A modified Markov Chain Monte Carlo (MCMC) approach based on Hit-and-Run sampling (Abbasi-Yadkori et al., 2017), adapted to boundary and corners exploration (details are in Appendix K).

For the above goals, we first design a diagnostic experiment using synthetic 2D Gaussian Mixture Models (GMMs) to isolate and stress-test key variables, and then evaluate QuiCK's practical capabilities on three real-world benchmarks spanning different modalities and label dimensions.

Evaluation Framework and Implementations. To comprehensively profile QuiCK, we consider five independent variables. In each controlled setting, we isolate one factor while holding the others constant: (1) Estimation Sample Size (n_s, n_t): Varies the amount of labeled source and unlabeled target data used to construct the weight confidence region $\boldsymbol{\Omega}$, thereby testing robustness to weight estimation uncertainty; (2) Calibration Sample Size (n): Evaluates algorithmic scalability and finite-sample threshold variance by scaling the size of the calibration set $\mathcal{D}_{\text{cali}}$; (3) Number of Classes (K): Stress-tests the dimensionality scaling. In this setting, we maintain a constant effective sample rate by fixing $n_s = n_t = 200 \times K$; (4) Nominal Target Coverage $(1 - \alpha)$: Assesses performance across varying prespecified prediction coverage levels; (5) Label Shift Intensity (Dirichlet Drift): Modulates the intensity of the distribution shift by tuning the Dirichlet concentration parameter. Following the above notation for concentration parameter, smaller values of a induce highly skewed marginal distributions, stress-testing the framework under extreme label shift scenarios.

For each experimental setting, we run 200 independent trials to ensure reproducibility, summarizing the evaluation results into two distinct blocks. The *Statistical Honesty* block measures statistical validity via Empirical Coverage and Average Prediction Set Size. The *Computational Efficiency* block measures algorithmic efficiency via end-to-end runtime and the number of LP evaluations triggered by Algorithm QuiCK.polytope. In all experiments reported below we set $\epsilon = 1/n$, which is far smaller than α in every setting considered.

4.1 Statistical Honesty Results

4.1.1 Prediction on Diagnostic Synthetic Datasets

Figure 1 presents how finite-sample errors in importance weight estimation have impact on conformal coverage by varying the estimation sample sizes n_s and n_t. The Oracle WCP perfectly achieves the exact 90.0% nominal level with a mean prediction set size of 5.54. The proposed frameworks, QuiCK.polytope and QuiCK.hyperbox, maintain strict and highly stable valid coverage (97.1%–97.2%) across all sample sizes. Both variants yield almost identical average set sizes ($\approx$ 7.75), demonstrating that QuiCK effectively encapsulates estimation uncertainty. By constructing rigorous confidence regions for the weights, QuiCK trades a marginal, controlled increase in set size for absolute statistical validity, even under extreme data scarcity ($n_s = n_t = 50$). In contrast, Plug-in WCP consistently fails to provide valid guarantees. This naive substitution leads to severe coverage degradation (86.2%–87.4%) and produces dangerous lower-tails, dropping below 80%, yielding artificially small set sizes (5.19–5.37). Finally, the sampling-based baseline, BBHR, exhibits severe sample-size dependency and high variance. It fails a bit to reach the nominal level at $n_s = n_t = 50$ (under-covering at 89.5%) and conservatively inflates both its coverage (93.9%) and set size (from 5.75 up to 6.57) as sample sizes increase, suggesting large samples are needed for consistent uncertainty quantification. Due to page limit, we show the impact of all other variables in Appendix M.

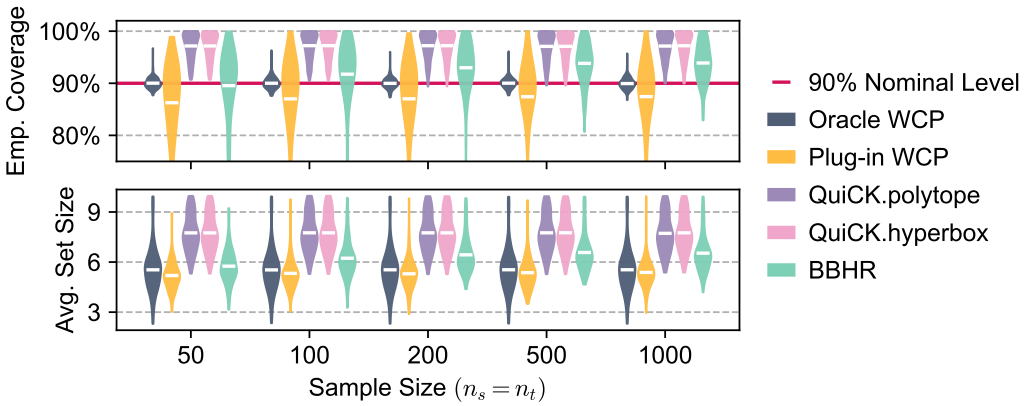


Figure 1: Empirical coverage and average prediction set size on synthetic datasets varying estimation sample sizes n_s, n_t ($K = 10$, 200 trials). Top: Empirical coverage relative to the 90% nominal level (magenta line). Bottom: Average prediction set size.

4.1.2 Prediction on Real-World Benchmarks

We evaluate the practical capability of our framework on real-world datasets, focusing our primary analysis on the large dimensional CIFAR-100 benchmark ($K = 100$).

We first evaluate the impact of estimation data scarcity under a severe, highly skewed label shift ($a = -2$). The Oracle WCP serves as the ideal baseline, maintaining a stable 90.0% nominal coverage with an average set size of ≈ 17.1 across all sample sizes. QuiCK strictly guarantees coverage regardless of estimation sample size. This evaluation directly corroborates our finite-sample uncertainty propagation mechanism: when estimation samples are smaller, the underlying Clopper-Pearson intervals are much wider. QuiCK faithfully propagates this statistical uncertainty, expanding prediction sets (up to ≈ 40.0) to conservatively protect the 90% coverage. As the estimation sample

size increases to 10,000, the input uncertainty naturally shrinks, and QuiCK.polytope converges its prediction sets down to 26.5. In this experimental setting, mean coverages for Plug-in WCP and BBHR appear stable, and no significant under-coverage happens.

For varying label shift intensities results, we show them in Appendix N.1. Meanwhile, comprehensive reports for the small dimensional FER2013 ($K = 7$) and 20 Newsgroups ($K = 20$) datasets show similar robust behaviors and are provided in Appendix N.2.

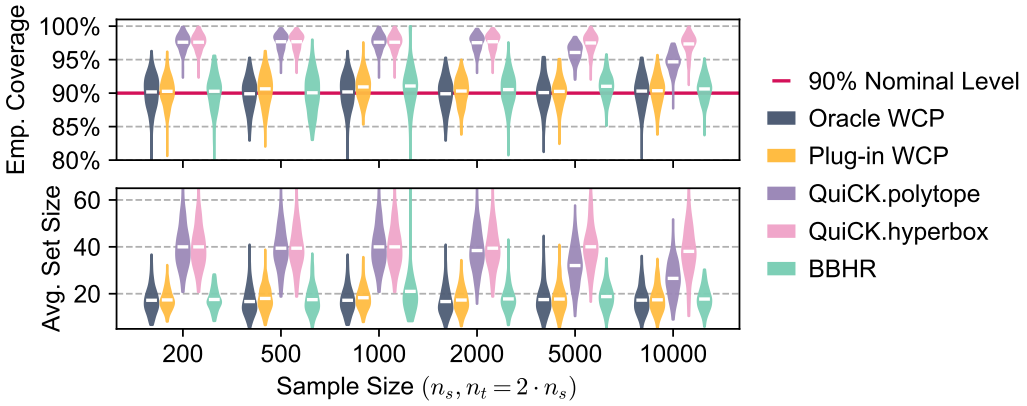


Figure 2: Empirical coverage and average prediction set size on the CIFAR-100 benchmark varying estimation sample sizes n_s and $n_t = 2 \cdot n_s$. The Dirichlet shift intensity is severely skewed at $a = -2$. Top: Empirical coverage. Bottom: Average prediction set size.

4.2 Computational Efficiency Results

Computational Statistics for Diagnostic Synthetic Datasets. After running the experiments, we found the most informative results are about label dimensions K and calibration sample sizes n. We place the end-to-end runtime and optimization statistics of QuiCK in Appendix N.3, including settings for across varying estimation sample sizes (n_s, n_t), and nominal coverages $(1 - \alpha)$.

The primary impact variable for QuiCK is the label dimensionality K. Table 2 and Table 3 show that, as K increases up to 200, the growing complexity naturally raises both average LP calls per unpruned class and solving times. However, benefiting from our dynamic class pruning mechanisms that achieves over 96% pruning rates when $K = 200$ (Table 3), QuiCK.polytope successfully scales to high-dimensional regimes, running significantly faster than the heuristic BBHR sampling. Notably, QuiCK.hyperbox is even faster than QuiCK.polytope when $K \leq 20$.

Set $K = 100$, Table 4 shows that QuiCK.polytope and QuiCK.hyperbox+ achieve robust scalability under varying calibration sizes n. As n increases from $5,000$ to $200,000$, runtime stays remarkably flat. This efficiency stems from QuiCK decoupling eCDF evaluation complexity as shown in Appendix G, nearly eliminating dependency on n. In contrast, Oracle WCP and Plug-in WCP+ are instantaneous (sub-millisecond) as they rely solely on deterministic point estimates without uncertainty propagation. Finally, the MCMC-based baseline, BBHR.polytope+, incurs the heaviest computational overhead by far due to its iterative sampling and burn-in phases.

Computational Statistics for Real-World Benchmarks. We evaluate the computational efficiency of QuiCK across three real-world datasets of varying dimensions, under variations in estimation sample sizes (n_s, n_t) and Dirichlet shift intensities (a). Across all real-world benchmarks, QuiCK demonstrates notable scalability and sub-second execution. On CIFAR-100 benchmark ($K = 100$), QuiCK.polytope completes end-to-end inference in merely 10–15 milliseconds, achieving more than $10\times$ speedup over the MCMC-based baseline BBHR.polytope+ ($\approx$ 180–200 ms). Meanwhile, QuiCK.polytope runs in near-instantaneous speed ($\approx$ 1.5 ms for 20 Newsgroups and $\approx$ 0.6 ms for FER2013). The detailed runtime and optimization statistics are summarized in Appendix N.4 (Tables 10–21).

AI USE STATEMENT

In this work, we used generative AI tools, `Gemini 3.6 Flash` and `Claude Sonnet 5`, for creating or modifying parts of the experimental code and language editing to improve readability. We have not used generative AI tools for developing theoretical models, formulating mathematical claims, assisting in proofs, proposing or refining hypotheses, designing or providing feedback on methodology or experiments, or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. We have carefully reviewed all AI-assisted work.

REFERENCES

Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, and Alan Malek. Hit-and-run for sampling and planning in non-convex spaces. In *Artificial intelligence and statistics*, pp. 888–895. PMLR, 2017.

Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *International conference on machine learning*, pp. 222–232. PMLR, 2020.

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*, 2019.

Aabesh Bhattacharyya and Rina Foygel Barber. Group-weighted conformal prediction. *Electronic Journal of Statistics*, 20(1):1171–1199, 2026.

Abraham Charnes and William W Cooper. Programming with linear fractional functionals. *Naval Research logistics quarterly*, 9(3-4):181–186, 1962.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Werner Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7):492–498, 1967.

Peter M Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and experience*, 24(3):327–336, 1994.

Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton. A unified view of label shift estimation. *Advances in Neural Information Processing Systems*, 33:3290–3300, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ying Jin, Zhimei Ren, and Emmanuel J Candès. Sensitivity analysis of individual treatment effects: A robust conformal inference approach. *Proceedings of the National Academy of Sciences*, 120 (6):e2214889120, 2023.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

Seong-ho Lee, Yanyuan Ma, and Jiwei Zhao. Efficient inference under label shift in unsupervised domain adaptation. *Journal of Machine Learning Research*, 27(193):1–50, 2026.

Jing Lei, Alessandro Rinaldo, and Larry Wasserman. A conformal prediction approach to explore functional data. *Annals of Mathematics and Artificial Intelligence*, 74(1):29–43, 2015.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113 (523):1094–1111, 2018.

Lihua Lei and Emmanuel J Candès. Conformal inference of counterfactuals and individual treatment effects. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(5):911–938, 2021.

Mushan Li, Kihyun Han, and Yanyuan Ma. From matrix inversion to constraints: Provably tighter confidence regions for importance weights in label shift. *arXiv preprint arXiv:2609.14802*, 2026.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pp. 3122–3130. PMLR, 2018.

Tom Mitchell. Twenty Newsgroups. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C5C323.

Sangdon Park, Edgar Dobriban, Insup Lee, and Osbert Bastani. PAC prediction sets under covariate shift. *arXiv preprint arXiv:2106.09848*, 2021.

Aleksandr Podkopaev and Aaditya Ramdas. Distribution-free uncertainty quantification for classification under label shift. In *Uncertainty in artificial intelligence*, pp. 844–853. PMLR, 2021.

Yaniv Romano, Matteo Sesia, and Emmanuel Candes. Classification with valid and adaptive coverage. *Advances in neural information processing systems*, 33:3581–3591, 2020.

Mauricio Sadinle, Jing Lei, and Larry Wasserman. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525):223–234, 2019.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

Siegfried Schaible. Fractional programming. In *Handbook of global optimization*, pp. 495–608. Springer, 1995.

Wenwen Si, Sangdon Park, Insup Lee, Edgar Dobriban, and Osbert Bastani. PAC prediction sets under label shift. *arXiv preprint arXiv:2310.12964*, 2023.

Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ioan M Stancu-Minasian. *Fractional programming: theory, methods and applications*, volume 409. Springer Science & Business Media, 2012.

Qinglong Tian, Xin Zhang, and Jiwei Zhao. Elsa: Efficient label shift adaptation through the lens of semiparametric models. In *International Conference on Machine Learning*, pp. 34120–34142. PMLR, 2023.

Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32, 2019.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005.

Mingzhang Yin, Claudia Shi, Yixin Wang, and David M Blei. Conformal sensitivity analysis for individual treatment effects. *Journal of the American Statistical Association*, 119(545):122–135, 2024.

Lutfiah Zahara, Purnawarman Musa, Eri Prasetyo Wibowo, Irwan Karim, and Saiful Bahri Musa. The facial emotion recognition (fer-2013) dataset for prediction system of micro-expressions face using the convolutional neural network (cnn) algorithm based raspberry pi. In *2020 Fifth international conference on informatics and computing (ICIC)*, pp. 1–9. IEEE, 2020.

A DISCUSSION

In this work, we addressed the computational tractability of honest conformal prediction under label shift. By decoupling class-specific weights from calibration-sample counts, QuiCK enables the simultaneous computation of conformal cutoffs for all classes. Beyond label shift adaptation, the technical machinery developed in QuiCK offers broader methodological implications for weighted conformal inference, for example, in group-weighted conformal prediction setting (Bhattacharyya & Barber, 2026). We extend this discussion in Appendix B. To handle massive number of labels, we found that processing the label Y on a superclass $Z = h(Y)$ under label shift is not a trivial problem. Because even when $p_{\mathbf{X}|Y} = q_{\mathbf{X}|Y}$, the induced class-conditional distributions $p_{\mathbf{X}|Z}$ and $q_{\mathbf{X}|Z}$ may differ whenever the within-superclass label proportion $p_{Y|Z} \neq q_{Y|Z}$. Moreover, when $p_{\mathbf{X}|Y}(\mathbf{x}, y)$ and $p_{\mathbf{X}|Y}(\mathbf{x}, y')$ are close for $h(y) = h(y') = z$, the induced difference between $p_{\mathbf{X}|Z}$ and $q_{\mathbf{X}|Z}$ is small, but $q_{Y|Z}$, required to identify $q_{\mathbf{X}|Z}$, is correspondingly difficult to recover from unlabeled target data. Conversely, when this $q_{Y|Z}$ is computable, the induced difference between $p_{\mathbf{X}|Z}$ and $q_{\mathbf{X}|Z}$ can be non-negligible. This tradeoff intertwines the $\mathbf{X} \mid Z$ and $Y \mid Z$ layers and makes it as a future study for QuiCK having a hierarchical extension.

B EXTENSION: GROUPED DISTRIBUTION SHIFT

The methods developed in this paper extend directly beyond label shift with $\mathcal{Y} = [K]$ to the case when the importance weights are shared within a finite number of groups. Let $\{\mathcal{A}_k\}_{k=1}^K$ be a known measurable partition of $\mathcal{X} \times \mathcal{Y}$, and let $\kappa(\mathbf{x}, y) = k$ when $(\mathbf{x}, y) \in \mathcal{A}_k$. We say that $\mathcal{P}$ and $\mathcal{Q}$ satisfy *grouped distribution shift* with respect to $\{\mathcal{A}_k\}_{k=1}^K$ if

$$\frac{q(\mathbf{x}, y)}{p(\mathbf{x}, y)} = \omega^*_{\kappa(\mathbf{x},y)}$$

for $\mathcal{P}$ almost every $(\mathbf{x}, y)$. Equivalently, the target to source density ratio is constant within each group, with

$$\omega^*_k = \frac{\mathrm{pr}_{(\mathbf{X},Y)\sim\mathcal{Q}}\{(\mathbf{X}, Y) \in \mathcal{A}_k\}}{\mathrm{pr}_{(\mathbf{X},Y)\sim\mathcal{P}}\{(\mathbf{X}, Y) \in \mathcal{A}_k\}}, \qquad k \in [K].$$

Thus, the group proportions may differ between the source and target populations, while the conditional distribution of $(\mathbf{X}, Y)$ within each group remains unchanged. Label shift is recovered by taking $\mathcal{A}_k = \mathcal{X} \times \{k\}$. Grouped covariate shift is obtained by partitioning $\mathcal{X}$ into $\{\mathcal{B}_k\}_{k=1}^K$ and taking $\mathcal{A}_k = \mathcal{B}_k \times \mathcal{Y}$.

For each $k \in [K]$, define

$$N_k(r) = \sum_{i=1}^n I\{(\mathbf{X}_i, Y_i) \in \mathcal{A}_k,\ r(\mathbf{X}_i, Y_i) \leq r\}, \qquad n_k = \sum_{i=1}^n I\{(\mathbf{X}_i, Y_i) \in \mathcal{A}_k\}.$$

For a candidate pair $(\mathbf{x}, y)$ with group index $j = \kappa(\mathbf{x}, y)$, let

$$n_k(j) = n_k + I\{j = k\}.$$

Then the weighted eCDF becomes

$$F(r; \boldsymbol{\omega}, j) = \frac{\sum_{k=1}^K N_k(r)\omega_k}{\sum_{k=1}^K n_k(j)\omega_k} = \frac{\sum_{k=1}^K N_k(r)\omega_k}{\sum_{k=1}^K n_k\omega_k + \omega_j}. \tag{3}$$

Equation (3) has exactly the same form as $F(r; \boldsymbol{\omega}, y)$ for label shift, which defines the weighted eCDF under label shift, with the class index replaced by the group index. Therefore, given a confidence region $\boldsymbol{\Omega}$ for the grouped importance weights, Theorem 1 (Honest conformal prediction under label shift) and all subsequent quantile optimization results extend directly to grouped distribution shift.

C ALGORITHMS

Algorithm QuiCK.hyperbox.base

Input: A hyperbox $1-\epsilon$ confidence region $\mathbf{\Omega} = \prod_{k=1}^{K}[\underline{\omega}_k, \overline{\omega}_k]$, $r(\mathbf{X}_i, Y_i)$ for $i = 1, ..., n$, prediction level $1-\alpha$.
Output: $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$ and t_0.
Procedure:
Sort $\{\tau_k\}_{k=1}^{K}$ so that $\tau_{\sigma(1)} \leq \cdots \leq \tau_{\sigma(K)}$.
for $t = 0, 1, ..., K-1$ **do**
  Set $\boldsymbol{\omega}^{(t)} = (\omega_1^{(t)}, ..., \omega_K^{(t)})$, where $\omega_{\sigma(k)} = \underline{\omega}_{\sigma(k)}$ for $k \leq K-t$, and $\omega_{\sigma(k)} = \overline{\omega}_{\sigma(k)}$ for $k > K-t$.
  if $\|\underline{\boldsymbol{\omega}}\|_2 = 0$ **then**
    continue ▷ Skip to next iteration
  end if
  Compute $\zeta(\boldsymbol{\omega}^{(t)})$.
  if $\zeta(\boldsymbol{\omega}^{(t)}) \geq \tau_{\sigma(K-t)}$ **then**
    Exit the **for** loop with $t_0 \leftarrow t$ and $\boldsymbol{\omega}^{\max} \leftarrow \boldsymbol{\omega}^{(t_0)}$.
  end if
end for
return $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(\boldsymbol{\omega}) = \zeta(\boldsymbol{\omega}^{\max})$ and t_0.

Algorithm QuiCK.hyperbox

Input: A hyperbox $1-\epsilon$ confidence region $\mathbf{\Omega} = \prod_{k=1}^{K}[\underline{\omega}_k, \overline{\omega}_k]$, $r(\mathbf{X}_i, Y_i)$ for $i = 1, ..., n$, prediction level $1-\alpha$.
Output: $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$ for $y = 1, ..., K$.
Procedure:
Apply **QuiCK.hyperbox.base** to get $(\tau_k)_{k=1}^{K}$ and t_0.
for $y = 1, ..., K$ **do**
  Take out τ_y and then insert $\tau_y(y)$ in $[\tau_{\sigma(1)}, ..., \tau_{\sigma(K)}]$ via Binary search to get $\tau_{\sigma_y(1)} \leq \cdots \leq \tau_{\sigma_y(K)}$. ▷ No need to re-sort all $\{\tau_k(y)\}_{k=1}^{K}$ for new y.
  for $t = 0, 1, ..., K$ **do**
    Set $\boldsymbol{\omega}^{(t)} = (\omega_1^{(t)}, ..., \omega_K^{(t)})$, where $\omega_{\sigma_y(k)} = \underline{\omega}_{\sigma_y(k)}$ for $k \leq K-t$, and $\omega_{\sigma_y(k)} = \overline{\omega}_{\sigma_y(k)}$ for $k > K-t$.
  end for
  if $\zeta(y;\boldsymbol{\omega}^{(t_0)}) \geq \tau_{\sigma_y(K-t_0)}(y)$ **then** ▷ Search $\boldsymbol{\omega}^{(t)}$ with $t \leq t_0$
    for $t = t_0, t_0-1, ..., 1$ **do**
      Compute $\zeta(y;\boldsymbol{\omega}^{(t-1)})$.
      if $\zeta(y;\boldsymbol{\omega}^{(t-1)}) < \tau_{\sigma_y(K-t+1)}(y)$ **then**
        Exit the **for** loop with $\boldsymbol{\omega}^{\max}(y) = \boldsymbol{\omega}^{(t)}$.
      end if
    end for
  else ▷ Search $\boldsymbol{\omega}^{(t)}$ with $t > t_0$
    for $t = t_0+1, t_0+2, ..., K$ **do**
      Compute $\zeta(y;\boldsymbol{\omega}^{(t)})$.
      if $\zeta(y;\boldsymbol{\omega}^{(t)}) \geq \tau_{\sigma_y(K-t)}(y)$ **then**
        Exit the **for** loop with $\boldsymbol{\omega}^{\max}(y) = \boldsymbol{\omega}^{(t)}$.
      end if
    end for
  end if
end for
return $\{\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})\}_{y=1}^{K}$.

Algorithm QuiCK.polytope.base

Input: A polytope $(1-\epsilon)$ confidence region $\mathbf{\Omega} = \{\boldsymbol{\omega} \in \mathbb{R}^K : A\boldsymbol{\omega} \leq \mathbf{b},\ \boldsymbol{\omega} \geq \mathbf{0}\}$, $r(\mathbf{X}_i, Y_i)$ for $i = 1, \ldots, n$, prediction level $1-\alpha$.
Output: $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(\boldsymbol{\omega})$.
Procedure:

Sort the $\{r(\mathbf{X}_i, Y_i)\}_{i=1}^n$ as $r_{(1)} \leq \cdots \leq r_{(n)}$.
Compute $n_k \leftarrow \sum_{i=1}^n I(Y_{(i)} = k)$ for $k = 1, \ldots, K$.
Initialize $l \leftarrow 1$, $u \leftarrow n$.
while $l \leq u$ **do**
    $m \leftarrow \lfloor (l+u)/2 \rfloor$.
    Compute $N_k(r_{(m)}) = \sum_{i=1}^n I\{Y_{(i)} = k, r_{(i)} \leq r_{(m)}\}$ for $k = 1, \ldots, K$.
    Solve $f_m \leftarrow \max_{\boldsymbol{\omega}\in\mathbf{\Omega}} \sum_{k=1}^K \{(1-\beta)n_k - N_k(r_{(m)})\}\omega_k$, recording the optimal $\boldsymbol{\omega}^{(m)}$ and the reduced costs $\check{c}_k(m)$, $k = 1, \ldots, K$.
    if $f_m > 0$ **then**
        $l \leftarrow m + 1$.
    else
        $u \leftarrow m - 1$.
    end if
end while
$l_0 \leftarrow l$
Compute $\boldsymbol{\omega}^{(l_0)}$ and $\{\check{c}_k(l_0)\}_{k=1}^K$.
return $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(\boldsymbol{\omega}) = r_{(l_0)}$.

D CHARACTERIZATION OF THE OPTIMAL IMPORTANCE WEIGHTS

We now characterize the optimal importance weights $\boldsymbol{\omega}^{\max}(y)$ that maximize $\zeta(y;\boldsymbol{\omega})$ over $\mathbf{\Omega}$. While computing the maximizer of $\zeta(y;\boldsymbol{\omega})$ is geometrically intractable, we use the fact that $\boldsymbol{\omega}^{\max}(y)$ can be the minimizer of $F(r;\boldsymbol{\omega},y)$ for some r (See Algorithm 5 in Appendix H). Accordingly, if we characterize a subset of $\mathbf{\Omega}$ that contains the minimizer of $F(r;\boldsymbol{\omega},y)$ for any r, y, then we can narrow down the search for $\boldsymbol{\omega}^{\max}(y)$ to that set.

First, we investigate, for fixed r, y, how $F(r;\boldsymbol{\omega},y)$ changes as $\boldsymbol{\omega}$ varies over $\mathbf{\Omega}$. Proposition 1 shows that constant multiplication of $\boldsymbol{\omega}$ does not affect the CDF $F(r;\boldsymbol{\omega},y)$, hence it also does not affect its $1-\beta$-quantile, *i.e.*, $\zeta(y;\boldsymbol{\omega}) = \zeta(y;c\boldsymbol{\omega})$. Proposition 1 is a direct result from (2).

Proposition 1 (Scale invariance)**.** *For $c > 0$ and $\boldsymbol{\omega} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$, $F(r;c\boldsymbol{\omega},y) = F(r;\boldsymbol{\omega},y)$ and $\zeta(y;\boldsymbol{\omega}) = \zeta(y;c\boldsymbol{\omega})$.*

Also, we consider changing $\boldsymbol{\omega}$ by a vector $\mathbf{v} \in \mathbb{R}^K$ and analyze how $F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y)$ changes with $\lambda \in \mathbb{R}$. Proposition 2 shows that $F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y)$ is a monotone function in λ as long as $\boldsymbol{\omega}+\lambda\mathbf{v} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$.

Proposition 2 (Monotonicity along a vector)**.** *Let $\boldsymbol{\omega} \equiv (\omega_1, ..., \omega_K) \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$ and $\mathbf{v} \equiv (v_1, ..., v_K)$ satisfy $\boldsymbol{\omega}+\mathbf{v} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}$.*

(i) If $F(r;\boldsymbol{\omega}+\mathbf{v},y) > F(r;\boldsymbol{\omega},y)$, then $F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y)$ is an increasing function of λ on $\{\lambda : \boldsymbol{\omega}+\lambda\mathbf{v} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}\}$.

(ii) If $F(r;\boldsymbol{\omega}+\mathbf{v},y) < F(r;\boldsymbol{\omega},y)$, then $F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y)$ is a decreasing function of λ on $\{\lambda : \boldsymbol{\omega}+\lambda\mathbf{v} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}\}$.

(iii) If $F(r;\boldsymbol{\omega}+\mathbf{v},y) = F(r;\boldsymbol{\omega},y)$, then $F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y) = F(r;\boldsymbol{\omega},y)$ for λ on $\{\lambda : \boldsymbol{\omega}+\lambda\mathbf{v} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\}\}$.

Let $\mathcal{S}(\mathbf{\Omega}) \equiv \{\boldsymbol{\omega} \in \mathbf{\Omega} : c\boldsymbol{\omega} \notin \mathbf{\Omega}^\circ \text{ for all } c > 0\}$, where $\mathbf{\Omega}^\circ$ denotes the interior of $\mathbf{\Omega}$. Let $\mathcal{E}(\mathbf{\Omega})$ denote the extreme points in $\mathbf{\Omega}$, *i.e.*,

$$\mathcal{E}(\mathbf{\Omega}) \equiv \{\boldsymbol{\omega} \in \mathbf{\Omega} : \text{for } \boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \in \mathbf{\Omega} \text{ with } \boldsymbol{\omega}_1 \neq \boldsymbol{\omega} \text{ and } \boldsymbol{\omega}_2 \neq \boldsymbol{\omega}, \boldsymbol{\omega} \neq \lambda\boldsymbol{\omega}_1 + (1-\lambda)\boldsymbol{\omega}_2,\ \forall\, \lambda \in (0,1)\}.$$

Algorithm QuiCK.polytope

Input: A polytope $(1-\epsilon)$ confidence region $\mathbf{\Omega} = \{\boldsymbol{\omega} \in \mathbb{R}_+^K \setminus \{\mathbf{0}\} : \mathbf{A}\boldsymbol{\omega} \leq \mathbf{b}, \boldsymbol{\omega} \geq \mathbf{0}\}$, $r(\mathbf{X}_i, Y_i)$ for $i = 1, \ldots, n$, prediction level $1-\alpha$.
Output: $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y; \boldsymbol{\omega})$ for $y = 1, \ldots, K$.
Procedure:

Apply **QuiCK.polytope.base** to get l_0, $\boldsymbol{\omega}^{(l_0)}$, and $\{\check{c}_k(l_0)\}_{k=1}^K$.
Set $r_{(n+1)} \leftarrow \infty$.
for $y = 1, \ldots, K$ **do**
  if $\omega_y^{(l_0)} = 0$ and $\check{c}_y(l_0) + (1-\beta) \leq 0$ **then** ▷ basis unchanged under $n_y \to n_y + 1$
    $l_0(y) \leftarrow l_0$. ▷ $f_{l_0}(y) = f_{l_0}$ unchanged
  else ▷ need to search $l \geq l_0$
    $l_0(y) \leftarrow n+1$. ▷ default: no crossing in $[l_0, n] \Rightarrow \zeta(y; \mathbf{\Omega}) = \infty$
    $step \leftarrow 0$, $m \leftarrow l_0$.
    Solve $f_m(y) \leftarrow \max_{\boldsymbol{\omega}\in\mathbf{\Omega}} \left[\sum_{k=1}^K \{(1-\beta)n_k(y) - N_k(r_{(m)})\}\omega_k\right]$.
    while $f_m(y) > 0$ and $m < n$ **do** ▷ exponential search
      $step \leftarrow \max(1, 2 \cdot step)$; $m \leftarrow \min(m + step,\ n)$.
      Solve $f_m(y)$ as above.
    end while
    if $f_m(y) \leq 0$ **then** ▷ binary search inside the bracket found above
      $l \leftarrow \max(l_0,\ m - step)$, $u \leftarrow m$.
      while $l < u$ **do**
        $m \leftarrow \lfloor (l+u)/2 \rfloor$.
        Solve $f_m(y)$ as above.
        if $f_m(y) > 0$ **then**
          $l \leftarrow m + 1$.
        else
          $u \leftarrow m$.
        end if
      end while
      $l_0(y) \leftarrow u$.
    end if
  end if
end for
return $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y; \boldsymbol{\omega}) = r_{(l_0(y))}$ for $y = 1, ..., K$.

Then Theorem 6 shows that the minimizer of $F(r; \boldsymbol{\omega}, y)$ always exists in $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$, and thus, $\boldsymbol{\omega}^{\max}(y)$ can be chosen from $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$.

Theorem 6. *For any fixed r, there exists $\boldsymbol{\omega}_0 \in \operatorname{argmin}_{\boldsymbol{\omega}\in\mathbf{\Omega}} F(r; \boldsymbol{\omega}, y)$ such that $\boldsymbol{\omega}_0 \in \mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$. In other words, it is sufficient to search $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$ to find $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y; \boldsymbol{\omega})$.*

We can observe in practice that $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$ narrows down the number of points $\boldsymbol{\omega}^{\max}(y)$ to search over. In particular, when $K = 2$, the solution $(\omega_1, \omega_2) \in \mathcal{S}(\mathbf{\Omega})$ should be either the maximizer or the minimizer of the ratio ω_1/ω_2 in $\mathbf{\Omega}$. Figure 3 shows the illustrations of $\mathcal{S}(\mathbf{\Omega})$ and $\mathcal{E}(\mathbf{\Omega})$ and demonstrates Theorem 6 that $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y; \boldsymbol{\omega})$ can be found in $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$. When $K = 2$ (left panel), $\mathcal{S}(\mathbf{\Omega}) = \{B, D\}$ and $\mathcal{E}(\mathbf{\Omega}) = \{A, B, C, D\}$. Then $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega}) = \{B, D\}$, which contains $\boldsymbol{\omega}^{\max} = B$. When $K = 3$ (right panel), $\mathcal{S}(\mathbf{\Omega})$ consists of six edges GH, FH, BF, AB, AC, and CH, and $\mathcal{E}(\mathbf{\Omega})$ is the set of eight vertices A–H. Then $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega}) = \{A, B, C, F, G, H\}$, which contains $\boldsymbol{\omega}^{\max} = F$.

E Related Work

Conformal prediction under distribution shift Tibshirani et al. (2019) introduced weighted conformal prediction for covariate shift. They reweight the calibration scores using the density ratio between the target and source covariate distributions to obtain valid marginal coverage in finite samples. Podkopaev & Ramdas (2021) applied this idea to label shift, where observations with label y share the weight $\omega_y^* = q_Y(y)/p_Y(y)$. Their coverage guarantee holds with the true weights. For es-

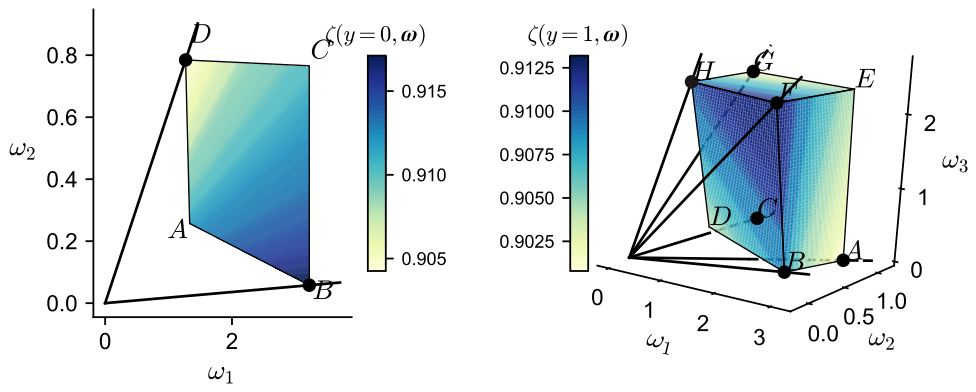


Figure 3: Illustrations of $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$. For $K = 2$ (left panel), when $\mathbf{\Omega}$ is the colored quadrangle, $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$ consists of two vertices, B and D. For $K = 3$ (right panel), when $\mathbf{\Omega}$ is the colored polytope, $\mathcal{S}(\mathbf{\Omega}) \cap \mathcal{E}(\mathbf{\Omega})$ consists of the six vertices A, B, C, F, G, H.

timated weights, they establish asymptotic validity under consistency and an additional condition on the weighted quantile. Bhattacharyya & Barber (2026) consider covariate shifts caused by changes in group proportions. By using the fact that observations in the same group share a weight, they obtain coverage bounds depending on the number of observations in each group. Our work uses the shared weights under label shift to compute $\sup_{\boldsymbol{\omega} \in \mathbf{\Omega}} \zeta(y; \boldsymbol{\omega})$ over a confidence region for the unknown class weights.

Estimation and inference under label shift The importance weights can be estimated from labeled source data and unlabeled target data. Lipton et al. (2018) introduced black box shift estimation, which uses the source confusion matrix and the target proportions of predicted labels to estimate the weights through inversion of confusion matrix. Azizzadenesheli et al. (2019) proposed a regularized estimator that accounts for uncertainty in the weights when the available samples are small. Another approach estimates the target class proportions by maximizing a likelihood based on classifier probabilities (Saerens et al., 2002; Alexandari et al., 2020). Tian et al. (2023) use semiparametric theory to construct estimating equations for the importance weights. Their ELSA estimator is consistent and asymptotically normal, providing a basis for asymptotic inference. Lee et al. (2026) study efficient inference for general parameters of the target population through estimation of the outcome density ratio. All these results concern asymptotic validity, rather than confidence regions with guaranteed coverage in finite samples.

For inference in finite samples, Si et al. (2023) construct confidence intervals for the entries of the source confusion matrix and the target proportions of predicted labels and propagate these intervals to the importance weights using interval Gaussian elimination. Li et al. (2026) instead retain the underlying linear relations as constraints, obtaining a tighter joint confidence region and computing marginal intervals by linear programming. These joint and marginal constructions provide the polytopes and hyperboxes considered in our paper (see Appendix F for details).

Predictive coverage gap with uncertain importance weights Lei & Candès (2021) quantify the effect of estimating importance weights on conformal coverage under covariate shift. For the true weight $\omega^*(\mathbf{x}) = q_{\mathbf{X}}(\mathbf{x})/p_{\mathbf{X}}(\mathbf{x})$ and an estimator $\widehat{\omega}(\mathbf{x})$, they show that the coverage gap is bounded by $E_{\mathbf{X} \sim \mathcal{P}_{\mathbf{X}}}\{|\widehat{\omega}(\mathbf{X}) - \omega^*(\mathbf{X})|\}/2$, which depends on the unknown true weights. Jin et al. (2023) construct robust conformal prediction sets for sensitivity analysis of individual treatment effects, allowing the target to source likelihood ratio $\omega^*(\mathbf{x}, y)$ to vary between unknown lower and upper bounds $l(\mathbf{x})$ and $u(\mathbf{x})$. When these bounds are estimated by $\widehat{l}(\mathbf{x})$ and $\widehat{u}(\mathbf{x})$, their coverage gap depends on the extent to which the estimated bounds fail to contain the true likelihood ratio.

Instead of computing the coverage gap, PAC prediction sets control the conditional coverage gap with high probability over the randomness on the calibration set, whereas marginal coverage guarantees averaging over these sources of randomness. Park et al. (2021) construct PAC prediction sets

under covariate shift using the confidence region for the true weights. Si et al. (2023) adapt this approach to label shift using simultaneous confidence intervals for the importance weights.

Worst-case quantile computation and optimization For fixed r and y, the weighted eCDF $F(r; \boldsymbol{\omega}, y)$ is a ratio of linear functions of $\boldsymbol{\omega}$. Charnes & Cooper (1962) showed how to transform the optimization of such a ratio into a linear program when the denominator is positive over the feasible region. Dinkelbach (1967) developed an iterative method that replaces the ratio with objectives obtained by subtracting a multiple of the denominator from the numerator. These methods provide standard approaches to minimizing the weighted eCDF over a polytope, as illustrated in Appendix H.

Jin et al. (2023) and Yin et al. (2024) study the maximization of weighted conformal quantiles under individual bounds on each observation's weight $\omega(X_i) \in [l(X_i), u(X_i)]$ for $i \in [n]$. They provide optimization procedures based on sorted calibrated scores. However, under label shift, observations in the same class must share a weight, even when their scores lie on opposite sides of the threshold, which changes the structure of the optimization problem.

For label shift, Li et al. (2026) report exact quantile maximization over marginal hyperboxes and use Monte Carlo sampling to approximate the supremum over the joint confidence region. Taking the largest quantile among sampled weight vectors gives a lower bound on the supremum, but a finite collection of sampled weights need not attain it.

F Construction of Confidence Region for Importance Weights

We summarize the finite-sample confidence-region construction of Li et al. (2026). Let $\boldsymbol{\gamma} : \mathcal{X} \to [0,1]^K$ be the soft-max classifier trained on $\mathcal{D}_{\text{train}}$, where $\boldsymbol{\gamma}(\mathbf{x}) = \{g_k(\mathbf{x})\}_{k=1}^K$. Define the hard-max classifier $g : \mathcal{X} \to [K]$ as $g(\mathbf{x}) = \arg\max_{k\in[K]} g_k(\mathbf{x})$, with ties broken deterministically. Denote the joint confusion matrix for the source population as $\mathbf{C}^* = (C^*_{ij})_{i,j\in[K]}$ and the predicted label proportion vector in the target domain as $\mathbf{q}^* = (q^*_i)_{i\in[K]}$, defined as

$$\begin{aligned} C^*_{ij} &= \text{pr}_{(\mathbf{X},Y)\sim\mathcal{P}}\{g(\mathbf{X}) = i, Y = j\}, \\ q^*_i &= \text{pr}_{\mathbf{X}\sim\mathcal{Q}_\mathbf{X}}\{g(\mathbf{X}) = i\}. \end{aligned}$$

Under label shift, these quantities satisfy $\mathbf{C}^*\boldsymbol{\omega}^* = \mathbf{q}^*$. This identity allows uncertainty about the importance weights to be characterized through quantities estimable from labeled source data and unlabeled target data.

The counts in the source data and the target data corresponding to each C^*_{ij} and q^*_i follow multinomial distributions conditional on $\mathcal{D}_{\text{train}}$. The confidence sets $[\underline{\mathbf{C}}, \overline{\mathbf{C}}]$ and $[\underline{\mathbf{q}}, \overline{\mathbf{q}}]$ for $\mathbf{C}^*$ and $\mathbf{q}^*$, respectively, are then constructed as elementwise boxes, where $\underline{\mathbf{C}} = (\underline{C}_{ij})_{i,j\in[K]}$, $\overline{\mathbf{C}} = (\overline{C}_{ij})_{i,j\in[K]}$, $\underline{\mathbf{q}} = (\underline{q}_k)_{k\in[K]}$, and $\overline{\mathbf{q}} = (\overline{q}_k)_{k\in[K]}$ are the collections of endpoints satisfying

$$\text{pr}\{\mathbf{C}^* \in [\underline{\mathbf{C}}, \overline{\mathbf{C}}], \quad \mathbf{q}^* \in [\underline{\mathbf{q}}, \overline{\mathbf{q}}] \mid \mathcal{D}_{\text{train}}\} \geq 1 - \epsilon$$

Rather than propagating these intervals through matrix inversion, they collect all nonnegative weight vectors $\boldsymbol{\omega}$ compatible with some $\mathbf{C} \in [\underline{\mathbf{C}}, \overline{\mathbf{C}}]$ and $\mathbf{q} \in [\underline{\mathbf{q}}, \overline{\mathbf{q}}]$ satisfying $\mathbf{C}\boldsymbol{\omega} = \mathbf{q}$ and define

$$\boldsymbol{\Omega}_1 \equiv \left\{\boldsymbol{\omega} : \exists\, \mathbf{C} \in [\underline{\mathbf{C}}, \overline{\mathbf{C}}],\, \exists\, \mathbf{q} \in [\underline{\mathbf{q}}, \overline{\mathbf{q}}] \text{ such that } \mathbf{C}\boldsymbol{\omega} = \mathbf{q},\, \boldsymbol{\omega} \geq \mathbf{0}\right\}.$$

This joint region has the equivalent characterization as

$$\boldsymbol{\Omega}_1 = \left\{\boldsymbol{\omega} : \overline{\mathbf{C}}\boldsymbol{\omega} \geq \underline{\mathbf{q}}, \quad \underline{\mathbf{C}}\boldsymbol{\omega} \leq \overline{\mathbf{q}}, \quad \boldsymbol{\omega} \geq \mathbf{0}\right\}.$$

Then $\boldsymbol{\Omega}_1$ can be written as $\boldsymbol{\Omega}_1 = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega} \leq \mathbf{b}, \boldsymbol{\omega} \geq \mathbf{0}\}$ by taking

$$\mathbf{A} = \begin{pmatrix} \underline{\mathbf{C}} \\ -\overline{\mathbf{C}} \end{pmatrix} \in \mathbb{R}^{2K\times K}, \qquad \mathbf{b} = \begin{pmatrix} \overline{\mathbf{q}} \\ -\underline{\mathbf{q}} \end{pmatrix} \in \mathbb{R}^{2K}.$$

Thus, the region is described directly by linear constraints and is thus a linear polytope. Whenever the elementwise intervals simultaneously cover $\mathbf{C}^*$ and $\mathbf{q}^*$, the identity $\mathbf{C}^*\boldsymbol{\omega}^* = \mathbf{q}^*$ ensures that $\boldsymbol{\omega}^* \in \boldsymbol{\Omega}_1$. Consequently, $\text{pr}\{\boldsymbol{\omega}^* \in \boldsymbol{\Omega}_1 \mid \mathcal{D}_{\text{train}}\} \geq 1 - \epsilon$.

Another confidence region can be obtained by solving $2K$ linear programs

$$\underline{\omega}_k = \min_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \omega_k, \qquad \overline{\omega}_k = \max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \omega_k, \qquad k \in [K].$$

The resulting $\boldsymbol{\Omega}_2 = \prod_{k=1}^{K} [\underline{\omega}_k, \overline{\omega}_k]$ is the smallest axis-aligned hyperbox containing $\boldsymbol{\Omega}_1$ and inherits its finite-sample coverage as a confidence region for $\boldsymbol{\omega}^*$. These constructions provide the confidence regions in the form of a polytope and a hyperbox considered in our paper, where the computational objective is to evaluate $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega})$ over the chosen region.

G WEIGHTED ECDF QUERIES

Both QuiCK.hyperbox and QuiCK.polytope repeatedly evaluate quantities involving the weighted eCDF at candidate score values while the weight vector $\boldsymbol{\omega}$ changes throughout the optimization. Evaluating these quantities directly from the calibration sample would require repeatedly scanning the n observations. We avoid this cost by separating the quantities that depend on $\boldsymbol{\omega}$ from the class-specific counts that depend only on the calibration data.

Let $r_{(1)} \le \cdots \le r_{(n)}$ denote the ordered calibration scores, and let $Y_{(i)}$ be the class corresponding to $r_{(i)}$. For each candidate score $r_{(t)}$, recall that for $k \in [K]$,

$$N_k(r_{(t)}) = \sum_{i=1}^{n} I\{Y_{(i)} = k, r_{(i)} \le r_{(t)}\}.$$

Let $\mathbf{N}(r_{(t)}) \equiv \{N_1(r_{(t)}), \ldots, N_K(r_{(t)})\}^{\mathrm{T}}$ and $\mathbf{n} \equiv (n_1, \ldots, n_K)^{\mathrm{T}}$. Then

$$F(r_{(t)};\boldsymbol{\omega}) = \frac{\mathbf{N}(r_{(t)})^{\mathrm{T}}\boldsymbol{\omega}}{\mathbf{n}^{\mathrm{T}}\boldsymbol{\omega}}, \qquad F(r_{(t)};\boldsymbol{\omega}, y) = \frac{\mathbf{N}(r_{(t)})^{\mathrm{T}}\boldsymbol{\omega}}{\mathbf{n}^{\mathrm{T}}\boldsymbol{\omega} + \omega_y}.$$

Thus, once $\mathbf{N}(r_{(t)})$ is available, evaluating either weighted eCDF requires only $O(K)$ arithmetic operations. Importantly, $\mathbf{N}(r_{(t)})$ does not depend on $\boldsymbol{\omega}$ or y and can therefore be precomputed or indexed once and reused throughout all subsequent optimization steps.

Several data structures can be used to obtain $\mathbf{N}(r_{(t)})$ efficiently.

- For each class k, store the calibration scores with $Y_i = k$ in sorted order. Since the full calibration scores have already been sorted, these class-specific lists can be constructed in $O(n)$ time. For a given $r_{(t)}$, $N_k(r_{(t)})$ is obtained by a binary search within the class-k list, requiring $O(\log n_k)$ time. Hence, obtaining all K counts costs $O\{\sum_{k=1}^{K} \log n_k\}$, which is $O\{K\log(n/K)\}$ when the class sizes are of comparable order. This method is referring to class-wise sorted index list.
- A Fenwick tree (Fenwick, 1994) indexed by the ordered calibration scores can store a K-vector of class counts at each node. A prefix-count query requires visiting $O(\log n)$ nodes and therefore costs $O(K\log n)$.
- A dense $K \times n$ prefix-count array can directly store $N_k(r_{(t)})$ for every $k \in [K]$ and $t \in [n]$. This requires $O(nK)$ preprocessing time and memory, but retrieving $\mathbf{N}(r_{(t)})$ and evaluating the weighted eCDF costs only $O(K)$.

The choice among these structures gives a memory–computation tradeoff. The class-specific sorted lists require only $O(n)$ storage and reduce a weighted eCDF query from $O(n)$ to $O\{K\log(n/K)\}$ when the classes are approximately balanced, while the dense prefix-count array further reduces the query cost to $O(K)$ at the expense of $O(nK)$ storage. These precomputed counts are reused across different weight vectors $\boldsymbol{\omega}$, candidate classes y, and candidate thresholds throughout the QuiCK algorithms.

In addition, for finding a quantile, we have the following analysis. After sorting the class-specific calibration scores, evaluating $F(r;\boldsymbol{\omega})$ at a fixed r costs $O\{K\log(n/K)\}$ (e.g. using class-wise sorted index list), and comparing $F(r;\boldsymbol{\omega})$ with $1-\alpha$ requires $O(\log n)$ evaluations to find $\zeta(\boldsymbol{\omega}^{(t)})$. Thus, computing each $\zeta(\boldsymbol{\omega}^{(t)})$ costs $OK\log(n/K)\log n$ computations.

H MINIMIZING THE WEIGHTED ECDF WITH FRACTIONAL PROGRAMMING

For a convex polytope $\mathbf{\Omega}$, we use an iterative procedure to solve $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$. To this end, at the t-th iteration, given $\boldsymbol{\omega}^{(t)}$ we would like to find $\boldsymbol{\omega}^{(t+1)}$ so that $\zeta(y;\boldsymbol{\omega}^{(t+1)}) > \zeta(y;\boldsymbol{\omega}^{(t)})$, which is equivalently written as $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} < 1-\beta \le F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t)},y\}$. This inspires us to set

$$\boldsymbol{\omega}^{(t+1)} = \text{argmin}_{\boldsymbol{\omega}} F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\}. \tag{4}$$

Here, we choose to perform the update by working with $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\}$ instead of with $\zeta(y;\boldsymbol{\omega})$ directly because of easier computation. However, an associated cost is that (4) does not guarantee $\zeta(y;\boldsymbol{\omega}^{(t+1)}) > \zeta(y;\boldsymbol{\omega}^{(t)})$. To this end, we need to compute $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\}$. If it is indeed $< 1-\beta$, then the update is successful. If it is $\ge 1-\beta$, then we have already achieved the optimal $\boldsymbol{\omega}^{\max}(y)$. Below, we adapt the above general strategy and ensure that the updates are always successful and the iteration procedure always converges.

Following the above analysis and using (2), at the t-th iteration, we can rewrite (4) as

$$\min_{\boldsymbol{\omega}} \frac{\sum_{k=1}^K N_k\{\zeta(y;\boldsymbol{\omega}^{(t)});y\}\omega_k}{\sum_{k=1}^K n_k\omega_k} \quad \text{subject to} \quad \mathbf{A}\boldsymbol{\omega}\le\mathbf{b}, \quad \boldsymbol{\omega}\ge\mathbf{0}. \tag{5}$$

The objective function in (5) is a fraction of linear functions of $\boldsymbol{\omega}$, while the constraint is also linear in $\boldsymbol{\omega}$, there (5) is readily solved by fractional programming method (Schaible, 1995; Stancu-Minasian, 2012). For example, we can employ the Charnes–Cooper transformation (Charnes & Cooper, 1962) to solve (5). To this end, let $s = \{\sum_{k=1}^K n_k\omega_k\}^{-1}$ and $\rho_k = s\omega_k$ for $k = 1,\dots,K$. Then one solution to (5) can be obtained by solving

$$\min_{\boldsymbol{\rho},s} \sum_{k=1}^K n_k\rho_k F_k\{\zeta(y;\boldsymbol{\omega}^{(t)});y\} \quad \text{subject to} \quad \mathbf{A}\boldsymbol{\rho}\le s\mathbf{b}, \quad \boldsymbol{\rho}\ge\mathbf{0}, \quad \sum_{k=1}^K n_k\rho_k = 1, \quad s>0. \tag{6}$$

via linear programming.

We adopt the update from $\boldsymbol{\omega}^{(t)}$ to $\boldsymbol{\omega}^{(t+1)}$ by solving (4) via fraction programming, and verifying

$$F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} < 1-\beta. \tag{7}$$

This procedure continues until (7) is violated, at which point we know $\zeta(y;\boldsymbol{\omega}^{(t)})$ is already the solution. Note that the iteration must end at $t \le n$ because the possible values of $\zeta(y;\boldsymbol{\omega})$ are at the point masses in (1), i.e, $\zeta(y;\boldsymbol{\omega}) \in \{r(\mathbf{X}_1,Y_1),...,r(\mathbf{X}_n,Y_n),\infty\}$.

Algorithm 5 Solve $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$ with a convex polytope $\mathbf{\Omega}$.

Input: A convex confidence region $\mathbf{\Omega}$, $r(\mathbf{X}_i,Y_i)$ for $i=1,...,n$.
Output: $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$.
Procedure:
Compute $\tau_k(y)$ for $k=1,...,K$
Take an initial point $\boldsymbol{\omega}^{(0)} \in \mathbf{\Omega}$.
for $t=0,\dots,n$ **do**
  Compute $\zeta(y;\boldsymbol{\omega}^{(t)})$ and $F_k\{\zeta(y;\boldsymbol{\omega}^{(t)});y\}$ for $k=1,....,K$,
  Apply fractional programming to obtain $\boldsymbol{\omega}^{(t+1)} = \text{argmin}_{\boldsymbol{\omega}\in\mathbf{\Omega}} F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\}$.
  if $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} \ge 1-\beta$ **then** ▷ $\zeta(y;\boldsymbol{\omega}^{(t)})$ will not increase further
    Exit the **for** loop with $\boldsymbol{\omega}^{\max}(y) = \boldsymbol{\omega}^{(t)}$.
  end if
end for
Set $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega}) = \zeta(y;\boldsymbol{\omega}^{\max}(y))$.

Theorem 7. *Let $\mathbf{\Omega} = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega}\le\mathbf{b}, \boldsymbol{\omega}\ge\mathbf{0}\}$ be a convex polytope. Then Algorithm 5 terminates in at most $n+1$ iterations and returns $\sup_{\boldsymbol{\omega}\in\mathbf{\Omega}} \zeta(y;\boldsymbol{\omega})$.*

Proof. At iteration t, the update $\boldsymbol{\omega}^{(t+1)} = \text{argmin}_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\}$ minimizes $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\}$ over $\boldsymbol{\Omega}$, so $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} \le F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t)},y\}$.

Consider the exit condition. If $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} \ge 1-\beta$, then since $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\}$ is the minimum over all $\boldsymbol{\omega} \in \boldsymbol{\Omega}$, we have $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega},y\} \ge 1-\beta$ for every $\boldsymbol{\omega} \in \boldsymbol{\Omega}$. By the definition of the $(1-\beta)$-quantile, this means $\zeta(y;\boldsymbol{\omega}) \le \zeta(y;\boldsymbol{\omega}^{(t)})$ for all $\boldsymbol{\omega} \in \boldsymbol{\Omega}$, so the algorithm correctly returns $\zeta(y;\boldsymbol{\omega}^{(t)})$.

Otherwise, we have $F\{\zeta(y;\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t+1)},y\} < 1-\beta$, which gives $\zeta(y;\boldsymbol{\omega}^{(t+1)}) > \zeta(y;\boldsymbol{\omega}^{(t)})$ by the definition of the quantile. Because $\zeta(y;\boldsymbol{\omega})$ takes values only in the finite set $\{r(\mathbf{X}_1,Y_1),\ldots,r(\mathbf{X}_n,Y_n),\infty\}$, which follows from the point-mass form of F in (1), a strictly increasing sequence can visit at most $n+1$ distinct values. Hence the loop terminates in at most $n+1$ iterations, and at termination the exit condition guarantees optimality. □

I FIGURES

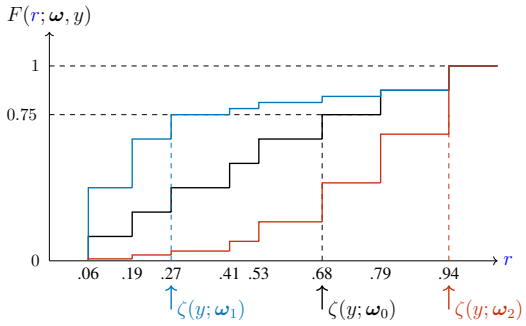


Figure 4: Weighted eCDFs under different weighting schemes. Horizontal axis shows $r(\mathbf{X}_i,Y_i)$ for $i = 1,..,8$, and 0.75-quantile of $F(\cdot;\boldsymbol{\omega},y)$ is indicated for each weighting schemes. (Black) Uniform weights $\boldsymbol{\omega}_0 = \mathbf{1}_K$ give $\zeta(y;\boldsymbol{\omega}_0) = 0.68$. (Blue) Left-concentrated weights $\boldsymbol{\omega}_1$ shift $\zeta(y;\boldsymbol{\omega})$ toward a smaller r, $\zeta(y;\boldsymbol{\omega}_1) = 0.27$. (Red) Right-skewed weights $\boldsymbol{\omega}_2$ push $\zeta(y;\boldsymbol{\omega})$ to a larger r, $\zeta(y;\boldsymbol{\omega}_2) = 0.94$.

J PROOFS

J.1 PROOF OF THEOREM 1

Proof. By Theorem 2 of Podkopaev & Ramdas (2021), $\text{pr}_{(\mathbf{X}_0,Y_0)\sim\mathcal{Q}}\{r(\mathbf{X}_0,Y_0) \le \zeta(Y_0;\boldsymbol{\omega}^*)\} \ge 1-\beta$. Then

$$
\begin{aligned}
& \text{pr}_{(\mathbf{X}_0,Y_0)\sim\mathcal{Q}}\{Y_0 \notin \mathcal{C}(\mathbf{X}_0;\boldsymbol{\Omega}) \mid \mathcal{D}_{\text{train}}\} \\
= \; & \text{pr}_{(\mathbf{X}_0,Y_0)\sim\mathcal{Q}}\{r(\mathbf{X}_0,Y_0) > \sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(Y_0;\boldsymbol{\omega}) \mid \mathcal{D}_{\text{train}}\} \\
\le \; & \text{pr}_{(\mathbf{X}_0,Y_0)\sim\mathcal{Q}}\{r(\mathbf{X}_0,Y_0) > \zeta(Y_0;\boldsymbol{\omega}^*) \mid \mathcal{D}_{\text{train}}\} + \text{pr}\{\boldsymbol{\omega}^* \notin \boldsymbol{\Omega} \mid \mathcal{D}_{\text{train}}\} \\
\le \; & \beta + \epsilon.
\end{aligned}
$$

□

J.2 Proof of Theorem 2

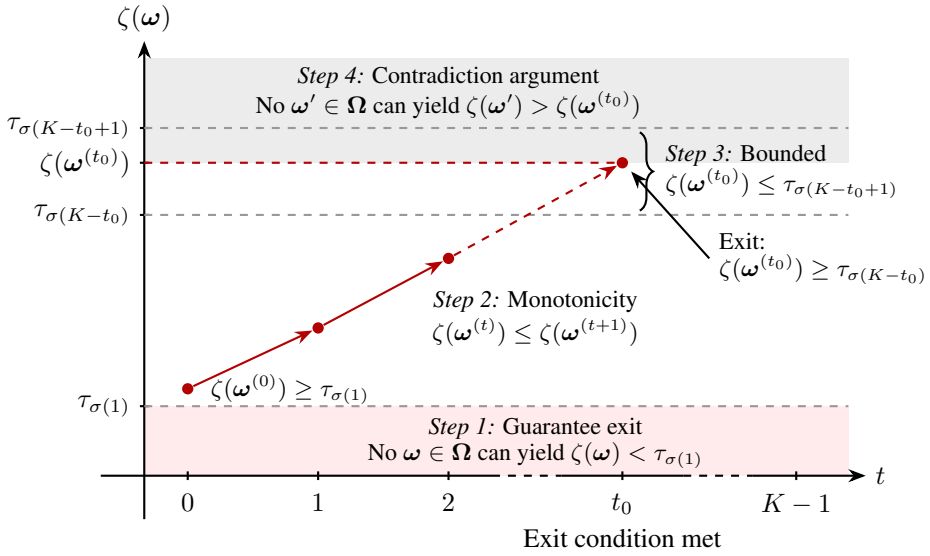


Figure 5: Roadmap of the proof for Theorem 2. The dark red sequence represents the evaluation of $\zeta(\boldsymbol{\omega}^{(t)})$ as t increases. The light red shaded region below shows that $\tau_{\sigma(1)}$ is strictly unreachable, guaranteeing algorithm exit. The proof consists of four main steps: (1) guaranteeing exit, (2) demonstrating monotonicity of ζ, (3) establishing upper bound at exit step t_0, and (4) proving global optimality by contradiction.

Proof. **Step 1.** We prove that it is impossible to have $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ such that $\zeta(\boldsymbol{\omega}) < \tau_{\sigma(1)}$. This is easily verified because $N_k\{\tau_{\sigma(1)}-\} < (1-\beta)n_k$ for $k \in [K]$ and thus,

$$\begin{aligned} F\{\tau_{\sigma(1)}-;\boldsymbol{\omega}\} &= \frac{\sum_{k=1}^{K} N_k\{\tau_{\sigma(1)}-\}\omega_k}{\sum_{k=1}^{K} n_k\omega_k} \\ &< \frac{\sum_{k=1}^{K} n_k(1-\beta)\omega_k}{\sum_{k=1}^{K} n_k\omega_k} \\ &= 1-\beta. \end{aligned}$$

Because $\zeta(\boldsymbol{\omega}^{(K-1)}) \ge \tau_{\sigma(1)}$, the **if** condition in Algorithm QuiCK.hyperbox.base will be satisfied for at least one $t \in \{0, 1, ..., K-1\}$, hence the algorithm will exit with an $\boldsymbol{\omega}^{\max}$ value. Let the exist t value be t_0, *i.e.*, $\zeta(\boldsymbol{\omega}^{(t_0)}) \ge \tau_{\sigma(K-t_0)}$ and $\zeta(\boldsymbol{\omega}^{(l)}) < \tau_{\sigma(K-l)}$ for all $l < t_0$.

Step 2. We show that $\zeta(\boldsymbol{\omega}^{(t)})$ is increasing with t for $t < t_0$, *i.e.*, $\zeta(\boldsymbol{\omega}^{(t)}) \le \zeta(\boldsymbol{\omega}^{(t+1)})$ for $t = 0, \ldots, t_0 - 1$. This is because when $t < t_0$, we have $\zeta(\boldsymbol{\omega}^{(t)}) < \tau_{\sigma(K-t)}$, hence

$$\begin{aligned} & F\{\zeta(\boldsymbol{\omega}^{(t)})-;\boldsymbol{\omega}^{(t+1)}\} \\ = & \frac{\sum_{k=1}^{K-t-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t)})-\}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t)})-\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\ = & \frac{\sum_{k=1}^{K-t} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t)})-\}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+1}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t)})-\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\ & + \frac{\overline{\omega}_{\sigma(K-t)} - \underline{\omega}_{\sigma(K-t)}}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} N_{\sigma(K-t)}\{\zeta(\boldsymbol{\omega}^{(t)})-\} \\ = & \frac{\sum_{k=1}^{K-t} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} F\{\zeta(\boldsymbol{\omega}^{(t)})-;\boldsymbol{\omega}^{(t)}\} \end{aligned}$$

$$
\begin{aligned}
&+\frac{n_{\sigma(K-t)}\big(\overline{\omega}_{\sigma(K-t)}-\underline{\omega}_{\sigma(K-t)}\big)}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}F_{\sigma(K-t)}\{\zeta(\boldsymbol{\omega}^{(t)})-\}\\
<\;& \frac{\sum_{k=1}^{K-t} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta)\\
&+\frac{n_{\sigma(K-t)}\big(\overline{\omega}_{\sigma(K-t)}-\underline{\omega}_{\sigma(K-t)}\big)}{\sum_{k=1}^{K-t-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta)\\
=\;& 1-\beta,
\end{aligned}
$$

i.e., $\zeta(\boldsymbol{\omega}^{(t)}) \le \zeta(\boldsymbol{\omega}^{(t+1)})$.

Step 3. We show $\zeta(\boldsymbol{\omega}^{(t_0)}) \le \tau_{\sigma(K-t_0+1)}$. This is because based on

$$
F\{\tau_{\sigma(K-t_0+1)};\boldsymbol{\omega}^{(t_0-1)}\} \ge F\{\zeta(\boldsymbol{\omega}^{(t_0-1)})\};\boldsymbol{\omega}^{(t_0-1)}\} \ge 1-\beta,
$$

we have

$$
\begin{aligned}
&F\{\tau_{\sigma(K-t_0+1)};\boldsymbol{\omega}^{(t_0)}\}\\
=\;& \frac{\sum_{k=1}^{K-t_0} N_{\sigma(k)}\{\tau_{\sigma(K-t_0+1)}\}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} N_{\sigma(k)}\{\tau_{\sigma(K-t_0+1)}\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t_0} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}\\
=\;& \frac{\sum_{k=1}^{K-t_0+1} N_{\sigma(k)}\{\tau_{\sigma(K-t_0+1)}\}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+2}^{K} N_{\sigma(k)}\{\tau_{\sigma(K-t_0+1)}\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t_0} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}\\
&+\frac{\overline{\omega}_{\sigma(K-t_0+1)}-\underline{\omega}_{\sigma(K-t_0+1)}}{\sum_{k=1}^{K-t_0} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}N_{\sigma(K-t_0+1)}\{\tau_{\sigma(K-t_0+1)}\}\\
=\;& \frac{\sum_{k=1}^{K-t_0+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t_0} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}F\{\tau_{\sigma(K-t_0+1)};\boldsymbol{\omega}^{(t_0-1)}\}\\
&+\frac{n_{\sigma(K-t_0+1)}\big(\overline{\omega}_{\sigma(K-t_0+1)}-\underline{\omega}_{\sigma(K-t_0+1)}\big)}{\sum_{k=1}^{K-t_0} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t_0+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}F_{\sigma(K-t_0+1)}\{\tau_{\sigma(K-t_0+1)}\}\\
\ge\;& 1-\beta,
\end{aligned}
$$

i.e., $\zeta(\boldsymbol{\omega}^{(t_0)}) \le \tau_{\sigma(K-t_0+1)}$.

Step 4. We prove by contradiction that for any other $\boldsymbol{\omega}' = (\omega'_1,...,\omega'_K) \in \boldsymbol{\Omega}$, $\zeta(\boldsymbol{\omega}') \le \zeta(\boldsymbol{\omega}^{(t_0)})$. Note that we have established $\tau_{\sigma(K-t_0)} \le \zeta(\boldsymbol{\omega}^{(t_0)}) \le \tau_{\sigma(K-t_0+1)}$. Let $m \equiv \min\{k \in [K] : \zeta(\boldsymbol{\omega}^{(t_0)}) < \tau_{\sigma(k)}\}$. Then $m \ge K-t_0+1$ and for $k \le m-1$, $\zeta(\boldsymbol{\omega}^{(t_0)}) \ge \tau_{\sigma(k)}$, for $k \ge m$, $\zeta(\boldsymbol{\omega}^{(t_0)}) < \tau_{\sigma(k)}$. Assume $\zeta(\boldsymbol{\omega}') > \zeta(\boldsymbol{\omega}^{(t_0)})$. Then

$$
\begin{aligned}
1-\beta \;>\;& F\{\zeta(\boldsymbol{\omega}^{(t_0)});\boldsymbol{\omega}'\}\\
=\;& \frac{\sum_{k=1}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\omega'_{\sigma(k)}}{\sum_{k=1}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}\\
=\;& \frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\underline{\omega}_{\sigma(k)}+\sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\omega'_{\sigma(k)}}{\sum_{k=1}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}\\
&+\frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}(\omega'_{\sigma(k)}-\underline{\omega}_{\sigma(k)})}{\sum_{k=1}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}\\
\ge\;& \frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\underline{\omega}_{\sigma(k)}+\sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\omega'_{\sigma(k)}}{\sum_{k=1}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}\\
&+\frac{\sum_{k=1}^{m-1} n_{\sigma(k)}(\omega'_{\sigma(k)}-\underline{\omega}_{\sigma(k)})(1-\beta)}{\sum_{k=1}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}},
\end{aligned}
$$

which leads to

$$1-\beta > \frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\omega'_{\sigma(k)}}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}. \tag{8}$$

Furthermore, for $k \geq m$, $F_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\} \leq F_{\sigma(k)}\{\tau_{\sigma(k)}-\} < 1-\beta$, so

$$N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\} < n_{\sigma(k)}(1-\beta). \tag{9}$$

Then we get

$$\begin{aligned}
& F\{\zeta(\boldsymbol{\omega}^{(t_0)}); \boldsymbol{\omega}^{(K-m+1)}\} \\
= \; & \frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\
= \; & \frac{\sum_{k=1}^{m-1} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}\omega'_{\sigma(k)}}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\
& + \frac{\sum_{k=m}^{K} N_{\sigma(k)}\{\zeta(\boldsymbol{\omega}^{(t_0)})\}(\overline{\omega}_{\sigma(k)} - \omega'_{\sigma(k)})}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\
< \; & \frac{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\omega'_{\sigma(k)}}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta) \\
& + \frac{\sum_{k=m}^{K} n_{\sigma(k)}(\overline{\omega}_{\sigma(k)} - \omega'_{\sigma(k)})}{\sum_{k=1}^{m-1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=m}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta) \\
= \; & 1-\beta,
\end{aligned}$$

where the inequality is due to (8) and (9). This implies $\zeta(\boldsymbol{\omega}^{(t_0)}) < \zeta(\boldsymbol{\omega}^{(K-m+1)})$.

However, $K-m+1 \leq t_0$ by the definition of m, this contradicts to the fact that $\zeta(\boldsymbol{\omega}^{(t)})$ is nondecreasing when $t < t_0$. Hence, $\boldsymbol{\omega}^{(t_0)} = \arg\max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(\boldsymbol{\omega})$. □

J.3 Proof of Theorem 3

Proof. The only thing to check is that

$$\zeta(y; \boldsymbol{\omega}^{(t)}) < \tau_{\sigma_y(K-t)}(y) \implies \zeta(y; \boldsymbol{\omega}^{(t-1)}) < \tau_{\sigma_y(K-t+1)}(y). \tag{10}$$

Then $\zeta(y; \boldsymbol{\omega}^{(t)}) \geq \tau_{\sigma_y(K-t)}(y)$ implies $\zeta(y; \boldsymbol{\omega}^{(t')}) \geq \tau_{\sigma_y(K-t')}(y)$ for all $t' > t$, and $\zeta(y; \boldsymbol{\omega}^{(t-1)}) < \tau_{\sigma_y(K-t+1)}(y)$ implies $\zeta(y; \boldsymbol{\omega}^{(t')}) < \tau_{\sigma_y(K-t')}(y)$ for all $t' < t$. Thus, the exit condition of Algorithm QuiCK.hyperbox, which is

$$\zeta(y; \boldsymbol{\omega}^{(t)}) \geq \tau_{\sigma_y(K-t)}(y), \quad \zeta(y; \boldsymbol{\omega}^{(t-1)}) < \tau_{\sigma_y(K-t+1)}(y),$$

gives that $\zeta(y; \boldsymbol{\omega}^{(t)})$ is the optimal value by Algorithm QuiCK.hyperbox.base, applied with y included. Now, (10) follows from Lemma 1 applied to the corresponding quantities including y. □

Lemma 1. *For $t = 1, ..., K-1$, $\zeta(\boldsymbol{\omega}^{(t)}) < \tau_{\sigma(K-t)}$ implies $\zeta(\boldsymbol{\omega}^{(t-1)}) < \tau_{\sigma(K-t+1)}$.*

Proof of Lemma 1. We prove $F\{\tau_{\sigma(K-t+1)}-; \boldsymbol{\omega}^{(t-1)}\} \geq 1-\beta$.

$$\begin{aligned}
& F\{\tau_{\sigma(K-t+1)}-; \boldsymbol{\omega}^{(t-1)}\} \\
= \; & \frac{\sum_{k=1}^{K-t+1} N_{\sigma(k)}\{\tau_{\sigma(K-t+1)}-\}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+2}^{K} N_{\sigma(k)}\{\tau_{\sigma(K-t+1)}-\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\
= \; & \frac{\sum_{k=1}^{K-t} N_{\sigma(k)}\{\tau_{\sigma(K-t+1)}-\}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+1}^{K} N_{\sigma(k)}\{\tau_{\sigma(K-t+1)}-\}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)} + \sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}
\end{aligned}$$

$$
\begin{aligned}
& -\frac{N_{\sigma(K-t+1)}\{\tau_{\sigma(K-t+1)}-\}(\overline{\omega}_{\sigma(K-t+1)}-\underline{\omega}_{\sigma(K-t+1)})}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} \\
= \quad & \frac{\sum_{k=1}^{K-t} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} F\{\tau_{\sigma(K-t+1)}-;\boldsymbol{\omega}^{(t)}\} \\
& -\frac{n_{\sigma(K-t+1)}(\overline{\omega}_{\sigma(K-t+1)}-\underline{\omega}_{\sigma(K-t+1)})}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}} F_{\sigma(K-t+1)}\{\tau_{\sigma(K-t+1)}-\}.
\end{aligned}
$$

Since $\zeta(\boldsymbol{\omega}^{(t)}) < \tau_{\sigma(K-t)} \le \tau_{\sigma(K-t+1)}$, we have

$$
F\{\tau_{\sigma(K-t+1)}-;\boldsymbol{\omega}^{(t)}\} \ge F\{\zeta(\boldsymbol{\omega}^{(t)});\boldsymbol{\omega}^{(t)}\} \ge 1-\beta.
$$

Also, by definition of $\tau_{\sigma(K-t+1)}$, we have

$$
F_{\sigma(K-t+1)}\{\tau_{\sigma(K-t+1)}-\} < 1-\beta.
$$

Combining these inequalities, we get

$$
\begin{aligned}
& F\{\tau_{\sigma(K-t+1)}-;\boldsymbol{\omega}^{(t-1)}\} \\
\ge \quad & \frac{\sum_{k=1}^{K-t} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+1}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta) \\
& -\frac{n_{\sigma(K-t+1)}(\overline{\omega}_{\sigma(K-t+1)}-\underline{\omega}_{\sigma(K-t+1)})}{\sum_{k=1}^{K-t+1} n_{\sigma(k)}\underline{\omega}_{\sigma(k)}+\sum_{k=K-t+2}^{K} n_{\sigma(k)}\overline{\omega}_{\sigma(k)}}(1-\beta) \\
= \quad & 1-\beta,
\end{aligned}
$$

which gives the desired result.

□

J.4 Proof of Theorem 4

Proof. For any candidate value $r_{(t)}$ with $t \in [n]$, $\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(\boldsymbol{\omega}) > r_{(t)}$ is equivalent to $f_t > 0$.

For each iteration of Algorithm QuiCK.polytope.base, the interval $[l, u]$ retains $j^\star$ while its length is halved. Starting from an interval of length n, after at most $\lceil \log_2(n+1) \rceil$, the algorithm terminates with $l = j^\star$ and $u = j^\star - 1$. Thus, the algorithm returns $r_{(l_0)} = r_{(j^\star)} = \sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(\boldsymbol{\omega})$. □

J.5 Proof of Theorem 5

Proof. For each $y \in [K]$, define

$$
f_t(y) \equiv \max_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} g_t(\boldsymbol{\omega}, y).
$$

As in the proof of Theorem 4, for any $t \in [n]$,

$$
\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \zeta(y;\boldsymbol{\omega}) > r_{(t)} \iff f_t(y) > 0,
$$

and $f_t(y)$ is nonincreasing in t. Hence, $l_0(y)$ is the smallest $t \in [n]$ such that $f_t(y) \le 0$, with $l_0(y) = n+1$ if $f_n(y) > 0$.

Since

$$
g_t(\boldsymbol{\omega}, y) = g_t(\boldsymbol{\omega}) + (1-\beta)\omega_y \ge g_t(\boldsymbol{\omega})
$$

for every $\boldsymbol{\omega} \in \boldsymbol{\Omega}$, we have $f_t(y) \ge f_t$. Therefore, for every $t < l_0$, $f_t > 0$ implies $f_t(y) > 0$, and hence $l_0(y) \ge l_0$.

Now suppose that $\omega_y^{(l_0)} = 0$ and

$$
\check{c}_y(l_0) + (1-\beta) \le 0.
$$

Since $1-\beta>0$, we have $\check{c}_y(l_0)<0$, so ω_y is nonbasic at the optimal solution of the base linear program. By linear programming sensitivity analysis, increasing the objective coefficient of ω_y by $1-\beta$ leaves the optimal basis unchanged. Because $\omega_y^{(l_0)}=0$, the optimal objective value is also unchanged, so

$$f_{l_0}(y)=f_{l_0}\le 0.$$

Together with $l_0(y)\ge l_0$, this gives $l_0(y)=l_0$.

If the sensitivity condition is not satisfied, Algorithm QuiCK.polytope searches sequentially over $t=l_0,l_0+1,\dots,n$ and exits at the first t such that $f_t(y)\le 0$. By the monotonicity of $f_t(y)$, this index is exactly $l_0(y)$. If no such $t\in[n]$ exists, then $f_n(y)>0$, so $l_0(y)=n+1$ and $r_{(l_0(y))}=r_{(n+1)}=\infty$.

Hence, for every $y\in[K]$, Algorithm QuiCK.polytope returns

$$r_{(l_0(y))}=\sup_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}\zeta(y;\boldsymbol{\omega}),$$

which proves the result. □

J.6 Proof of Proposition 2

Proof. Consider case (i). By (2),

$$F(r;\boldsymbol{\omega}+\lambda\mathbf{v},y)=\frac{\sum_{k=1}^K N_k(r)(\omega_k+\lambda v_k)}{\sum_{k=1}^K n_k(\omega_k+\lambda v_k)}.$$

For λ_1 and λ_2, $F(r;\boldsymbol{\omega}+\lambda_1\mathbf{v},y)>F(r;\boldsymbol{\omega}+\lambda_2\mathbf{v},y)$ is equivalent to

$$\begin{aligned}
& F(r;\boldsymbol{\omega}+\lambda_1\mathbf{v},y)>F(r;\boldsymbol{\omega}+\lambda_2\mathbf{v},y)\\
\iff\quad & \{\sum_{k=1}^K N_k(r)(\omega_k+\lambda_1 v_k)\}\{\sum_{l=1}^K n_l(\omega_l+\lambda_2 v_l)\}>\{\sum_{k=1}^K N_k(r)(\omega_k+\lambda_2 v_k)\}\{\sum_{l=1}^K n_l(\omega_l+\lambda_1 v_l)\}\\
\iff\quad & \lambda_1\sum_{k=1}^K\sum_{l=1}^K N_k(r)v_k n_l\omega_l+\lambda_2\sum_{k=1}^K\sum_{l=1}^K N_k(r)\omega_k n_l v_l\\
& >\lambda_1\sum_{k=1}^K\sum_{l=1}^K N_k(r)\omega_k n_l v_l+\lambda_2\sum_{k=1}^K\sum_{l=1}^K N_k(r)v_k n_l\omega_l\\
\iff\quad & (\lambda_1-\lambda_2)\sum_{k=1}^K\left\{\sum_{l=1}^K N_k(r)n_l\omega_l-\sum_{l=1}^K N_l(r)\omega_l n_k\right\}v_k>0.
\end{aligned}$$

Taking $\lambda_1=1$ and $\lambda_2=0$, we get

$$F(r;\boldsymbol{\omega}+\mathbf{v},y)>F(r;\boldsymbol{\omega},y)\iff\sum_{k=1}^K\left\{\sum_{l=1}^K N_k(r)n_l\omega_l-\sum_{l=1}^K N_l(r)\omega_l n_k\right\}v_k>0,$$

which is further equivalent to $F(r;\boldsymbol{\omega}+\lambda_1\mathbf{v},y)>F(r;\boldsymbol{\omega}+\lambda_2\mathbf{v},y)$ for all $\lambda_1>\lambda_2$. Hence, the desired result in case (i) holds.

Changing the inequality direction $>$ to $<$ and $=$ leads to the results in case (ii) and (iii), respectively. □

J.7 Proof of Theorem 6

Lemma 2. *Suppose that $F(r;\boldsymbol{\omega},y)\neq F(r;\boldsymbol{\omega}',y)$ for some $\boldsymbol{\omega},\boldsymbol{\omega}'\in\boldsymbol{\Omega}$. Then*

$$\textit{argmin}_{\boldsymbol{\omega}\in\boldsymbol{\Omega}}F(r;\boldsymbol{\omega},y)\mathcal{S}(\boldsymbol{\Omega}).$$

Proof of Lemma 2. Let $\boldsymbol{\omega}_0 \in \boldsymbol{\Omega}$ be a minimizer of $F(r;\cdot,y)$. Assume that $\boldsymbol{\omega}_0 \notin \mathcal{S}(\boldsymbol{\Omega})$, so that $c\boldsymbol{\omega}_0$ is in the interior of $\boldsymbol{\Omega}$ for some $c > 0$. Let $\boldsymbol{\omega}_1 \in \boldsymbol{\Omega}$ satisfy $F(r;\boldsymbol{\omega}_1,y) > F(r;\boldsymbol{\omega}_0,y)$. Then by Proposition 2, we can always find $\boldsymbol{\omega}_2 \in \boldsymbol{\Omega}$ on the line connecting $\boldsymbol{\omega}_1$ and $c\boldsymbol{\omega}_0$ such that

$$F(r;c\boldsymbol{\omega}_0,y) > F(r;\boldsymbol{\omega}_2,y)$$

by taking $\boldsymbol{\omega}_2 = c\boldsymbol{\omega}_0 - \lambda(\boldsymbol{\omega}_1 - c\boldsymbol{\omega}_0)$ with $\lambda > 0$. By Proposition 1, $F(r;\boldsymbol{\omega}_0,y) = F(r;c\boldsymbol{\omega}_0,y)$, so we get a contradiction that $\boldsymbol{\omega}_0$ is the minimizer of $F(r;\cdot,y)$ in $\boldsymbol{\Omega}$. Hence, $\boldsymbol{\omega}_0 \in \mathcal{S}(\boldsymbol{\Omega})$. □

Lemma 3. *There exists $\boldsymbol{\omega}_0 \in$ argmin$_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F(r;\boldsymbol{\omega},y)$ such that $\boldsymbol{\omega}_0 \in \mathcal{E}(\boldsymbol{\Omega})$.*

Proof of Lemma 3. Assume that there is no minimizer of $F(r;\cdot,y)$ in $\mathcal{E}(\boldsymbol{\Omega})$. Let $\boldsymbol{\omega}_0 = (\omega_{01},...,\omega_{0K}) \in \boldsymbol{\Omega}$ be a minimizer of $F(r;\cdot,y)$ with $\boldsymbol{\omega}_0 \notin \mathcal{E}(\boldsymbol{\Omega})$. We can represent $\boldsymbol{\omega}_0 = \sum_{m=1}^{M} \lambda_m \boldsymbol{\omega}_m$, where for each $m = 1,...,M$, $\boldsymbol{\omega}_m = (\omega_{m1},...,\omega_{mK}) \in \mathcal{E}(\boldsymbol{\Omega})$ and $\lambda_m > 0$ with $\sum_{m=1}^{M} \lambda_m = 1$.

Since $\boldsymbol{\omega}_0$ is the minimizer of $F(r;\cdot,y)$, we get that for each $m = 1,...,M$,

$$F(r;\boldsymbol{\omega}_m,y) = \frac{\sum_{k=1}^{K} N_k(r)\omega_{mk}}{\sum_{k=1}^{K} n_k\omega_{mk}} > F(r;\boldsymbol{\omega}_0,y) = \frac{\sum_{k=1}^{K} N_k(r)\omega_{0k}}{\sum_{k=1}^{K} n_k\omega_{0k}}.$$

Then for each $m = 1,...,M$, we have that

$$\sum_{k=1}^{K} N_k(r)\omega_{mk} \sum_{l=1}^{K} n_l\omega_{0l} > \sum_{k=1}^{K} N_k(r)\omega_{0k} \sum_{l=1}^{K} n_l\omega_{ml}.$$

Multiplying by λ_m and adding over $m = 1,...,M$ lead to

$$\sum_{m=1}^{M} \lambda_m \sum_{k=1}^{K} N_k(r)\omega_{mk} \sum_{l=1}^{K} n_l\omega_{0l} > \sum_{m=1}^{M} \lambda_m \sum_{k=1}^{K} N_k(r)\omega_{0k} \sum_{l=1}^{K} n_l\omega_{ml},$$

which is equivalent to

$$\frac{\sum_{k=1}^{K} N_k(r) \sum_{m=1}^{M} \lambda_m\omega_{mk}}{\sum_{k=1}^{K} n_k \sum_{m=1}^{M} \lambda_m\omega_{mk}} > \frac{\sum_{k=1}^{K} N_k(r)\omega_{0k}}{\sum_{k=1}^{K} n_k\omega_{0k}}.$$

By (2), the left-hand side equals $F(r;\sum_{m=1}^{M} \lambda_m\boldsymbol{\omega}_m,y) = F(r;\boldsymbol{\omega}_0,y)$, and the right-hand side equals $F(r;\boldsymbol{\omega}_0,y)$, which is a contradiction. □

Proof of Theorem 6. Suppose that $F(r;\boldsymbol{\omega},y) \neq F(r;\boldsymbol{\omega}',y)$ for some $\boldsymbol{\omega},\boldsymbol{\omega}' \in \boldsymbol{\Omega}$. By Lemma 3, there exists $\boldsymbol{\omega}_0 \in$ argmin$_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F(r;\boldsymbol{\omega},y) \cap \mathcal{E}(\boldsymbol{\Omega})$. Then by Lemma 2, argmin$_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F(r;\boldsymbol{\omega},y)\mathcal{S}(\boldsymbol{\Omega})$, so

$$\boldsymbol{\omega}_0 \in \text{argmin}_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F(r;\boldsymbol{\omega},y) \cap \mathcal{S}(\boldsymbol{\Omega}) \cap \mathcal{E}(\boldsymbol{\Omega}).$$

If $F(r;\boldsymbol{\omega},y) = F(r;\boldsymbol{\omega}',y)$ for all $\boldsymbol{\omega},\boldsymbol{\omega}' \in \boldsymbol{\Omega}$, then $\boldsymbol{\Omega} = \text{argmin}_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} F(r;\boldsymbol{\omega},y)$. Thus, any point $\boldsymbol{\omega}_0$ in $\mathcal{S}(\boldsymbol{\Omega}) \cap \mathcal{E}(\boldsymbol{\Omega})$ is the minimizer.

The last claim follows since $\boldsymbol{\omega}^{\max}(y)$ can be obtained as the minimizer of $F(r;\boldsymbol{\omega},y)$ for some r, given by Algorithm 5 □

K SAMPLING FROM THE POLYTOPE

We sample weights vectors from the polytope $\mathbf{\Omega} = \{\boldsymbol{\omega} : \mathbf{A}\boldsymbol{\omega} \leq \mathbf{b}\}$ using a boundary-biased variant of Hit-and-Run (Abbasi-Yadkori et al., 2017), which is a Markov Chain Monte Carlo (MCMC) method. The chain is initialized at the center of the largest Euclidean ball inscribed in $\mathbf{\Omega}$ (also known as the Chebyshev center). At each step, a random direction is drawn, the corresponding feasible chord inside $\mathbf{\Omega}$ is computed, and the next point is chosen along this chord from a $\mathrm{Beta}(0.1, 0.1)$ distribution. Note that small parameters in the Beta distribution yield a stronger sampling bias toward the boundary and vertices, effectively exploring the worst-case extreme points of the confidence region.

Algorithm Boundary-Biased Hit-and-Run (BBHR)

Input: Constraint matrix $\mathbf{A} \in \mathbb{R}^{3K \times K}$, vector $\mathbf{b} \in \mathbb{R}^{3K}$, number of samples N, burn-in period T_b
Output: Samples $\{\boldsymbol{\omega}^{(i)}\}_{i=1}^{N} \mathbf{\Omega}$
Procedure:

Compute the Chebyshev center of $\mathbf{\Omega}$ and set $\boldsymbol{\omega}^{(0)}$ to this point
for $t = 1, \ldots, T_b + N$ **do**
    Sample direction $\boldsymbol{d}$ by normalizing a standard Gaussian vector $\mathcal{N}(\mathbf{0}, \mathbf{I}_K)$
    Compute feasible step-size interval $[\lambda_{\min}, \lambda_{\max}]$ of the line $\boldsymbol{\omega}^{(t-1)} + \lambda \boldsymbol{d}$ in $\mathbf{\Omega}$
    Draw $u \sim \mathrm{Beta}(0.1, 0.1)$ ▷ Bias towards boundaries and vertices
    $\lambda \leftarrow \lambda_{\min} + u(\lambda_{\max} - \lambda_{\min})$
    $\boldsymbol{\omega}^{(t)} \leftarrow \boldsymbol{\omega}^{(t-1)} + \lambda \boldsymbol{d}$
end for
return $\{\boldsymbol{\omega}^{(T_b+1)}, \ldots, \boldsymbol{\omega}^{(T_b+N)}\}$

L EXPERIMENTAL DETAILS

We provide the complete setup for our diagnostic synthetic experiments and the comprehensive configurations for the real-world benchmarks, ensuring the reproducibility of all empirical results reported in the main text.

L.1 PLATFORM

Google Colab with NVIDIA H100 GPU was used to train/fine-tune the classifiers. The rest of the analysis was performed on a MacBook Pro with the Apple M3 Pro chip (11 CPU cores, 14-core GPU, and 18 GB of unified memory). The code is implemented in `Python 3.13` and `C++17` with environment management using `conda 24.11.3`.

L.2 DATA OVERVIEW

Diagnostic Synthetic Datasets. We draw samples from the Gaussian Mixture distributions with $K = 10$. The class-conditional feature distribution, $\mathbf{X} \mid Y = y$, is a standard multivariate Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}_y, \mathbf{I}_2)$, where the class means $\{\boldsymbol{\mu}_y\}_{y=1}^{10}$ are arranged on a regular grid around the origin. We use Uniform and Dirichlet priors to simulate the marginal label distributions. The detailed data-generating processes are placed in Appendix L.3.

Real-World Benchmarks. We benchmark QuiCK on FER2013 ($K = 7$, VGG19) (Zahara et al., 2020; Simonyan, 2014), 20 Newsgroups ($K = 20$, fine-tuned BERT) (Mitchell, 1997; Lang, 1995; Devlin et al., 2019), and CIFAR-100 ($K = 100$, ResNet18) (Krizhevsky et al., 2009; He et al., 2016). Across all real-world datasets, we enforce a strict disjoint data-splitting protocol. Base classifiers are trained exclusively on a dedicated *training partition*, while the separate *evaluation partition* is solely used for constructing weight estimators and computing conformal prediction sets alongside their empirical coverage. The exact data arrangements and classifier training parameters are detailed in Appendix L.4 and Table 1. Data generating process follows a multinomial stratified sampling procedure, and label shift settings are generated by varying the concentration parameter, where we denote it as 10^a, $a \in \{-3, -2, -1, 0, 1, 2, 3\}$. The detailed configurations are placed in Appendix L.5.

L.3 SYNTHETIC DATA GENERATION PROTOCOLS

General Data Generation Setup. Across the diagnostic experiments, the feature space is $\mathcal{X}\mathbb{R}^2$. For a K-class problem, the class-conditional distributions are identically parameterized as isotropic Gaussians, $p_{\mathbf{X}|Y}(\mathbf{x} \mid y) = \mathcal{N}(\boldsymbol{\mu}_y, \mathbf{I}_2)$. To systematically control class overlap without introducing arbitrary cluster shapes, the K class means $\{\boldsymbol{\mu}_y\}_{y=1}^{K}$ are arranged symmetrically on a $\lceil\sqrt{K}\rceil \times \lceil\sqrt{K}\rceil$ regular grid. The source population's marginal label distribution is generated from $\text{Dir}(6 \cdot \mathbf{1}_K)$ when investigating the Estimation Sample Size impact, and is fixed to a uniform distribution ($p_Y(y) = 1/K$ for all $y \in \mathcal{Y}$) for examining all other variables' impacts. To simulate severe label shift in the target population, the target marginal distribution q_Y is sampled from a Dirichlet distribution $\text{Dir}(0.1 \cdot \mathbf{1}_K)$. The true importance weights are intrinsically derived as $\omega_y^* = q_Y(y)/p_Y(y)$.

Specific Settings for Diagnostic Variables. Unless isolated as the independent testing variable in a specific setting, the default environment parameters are set to: Naive Bayes classifier with true parameters, $K = 100$, target nominal coverage $1 - \alpha = 0.90$, calibration sample size $n = 100,000$, and estimation sample sizes $n_s = n_t = 20,000$ (enforcing $200 \times K$). For the Monte Carlo sampling baseline (BBHR), we use a burn-in of $\min(100K, 10000)$ and sample $1,000$ weight vectors using a $\text{Beta}(0.1, 0.1)$ boundary-biased step size. The specific configurations for each stress-test variable are as follows:

- Estimation Sample Size (n_s, n_t): Fix $K = 10$, $n_s = n_t \in \{50, 100, 200, 500, 1000\}$;
- Calibration Sample Size (n): n is scaled as $c \times 10^3$, where $c \in \{5, 10, 20, 50, 100, 200\}$;

- Number of Classes (K): $K \in \{2, 5, 10, 20, 50, 100, 200\}$. We strictly maintain $n_s = n_t = 200 \times K$ to decouple dimensionality scaling from sample scarcity;
- Nominal Coverage ($1 - \alpha$): $1 - \alpha \in \{0.90, 0.95, 0.98, 0.99, 0.995\}$.

L.4 Real-World Benchmark Data Organization and Classifiers

To preserve the statistical honesty of the conformal coverage guarantees, we enforce a strict disjoint data-splitting protocol across all three real-world datasets. For each dataset, a massive, dedicated pool of samples is held out from the standard training set. This held-out *Evaluation Partition* ($\mathcal{D}_{\text{src}}, \mathcal{D}_{\text{tgt}}, \mathcal{D}_{\text{cali}}, \mathcal{D}_{\text{test}}$) is never exposed to the base classifier during training; it is reserved exclusively for constructing the weight estimator and computing conformal nonconformity scores.

The base classifiers are trained purely on the remaining *Training Partition*. To ensure robustness, we repeat the random data splitting, model training, and conformal prediction pipeline across 10 independent random seeds for each dataset. A summary of the dataset partitions, base model architectures, and training hyperparameters is provided in Table 1.

Table 1: Comprehensive configurations for real-world benchmarks. Base classifiers are exclusively trained on the *Training Partition*, leaving the *Evaluation Partition* strictly held-out for conformal calibration and weight estimation.

Dataset K, Format	Partitions (*Train* + *Eval.*)	Architecture	Training Hyperparameters & Augmentations
FER2013 $K = 7$ images	13,709 + 15,000	VGG19	Variant: with Batch Normalization Optimizer: SGD (lr=0.1, momentum=0.9, weight_decay=5e-4) Schedule: Cosine Annealing, 200 epochs, batch size 256 Augmentation: RandomCrop(48, pad=4), RandomHorizontalFlip
20 Newsgroups $K = 20$ documents	5,314 + 6,000	BERT	Variant: `bert-base-uncased` Optimizer: AdamW (lr=2e-5, weight_decay=0.01) Schedule: Linear Warmup (10% steps), 4 epochs, batch size 16 Preprocessing: Headers, footers, and quotes explicitly stripped
CIFAR-100 $K = 100$ images	20,000 + 30,000	ResNet18	Optimizer: SGD (lr=0.1, momentum=0.9, weight_decay=5e-4) Schedule: Cosine Annealing, 200 epochs, batch size 256 Augmentation: RandomCrop(32, pad=4), RandomHorizontalFlip

L.5 Real-World Benchmark Sampling and Label Shift Simulation

To systematically evaluate QuiCK under controlled label shift conditions on real-world datasets, we utilize the massive held-out *Evaluation Partition* as a data population. From this population, we dynamically construct the required subsets ($\mathcal{D}_{\text{cali}}$, $\mathcal{D}_{\text{src}}$, $\mathcal{D}_{\text{tgt}}$, and $\mathcal{D}_{\text{test}}$) using a multinomial stratified sampling procedure.

Multinomial Stratified Sampling Procedure. Specifically, given a desired total sample size c and a designated label proportion vector $\mathbf{v}$ (either the source p_Y or the target q_Y), we first sample a class count vector $\mathbf{c} \sim \text{Multinomial}(c, \mathbf{v})$. Then, for each class $y \in \mathcal{Y}$, we draw exactly c_y instances from the evaluation pool. This mathematical procedure ensures that the empirical class counts in the constructed subsets strictly conform to the expected marginal distributions, eliminating the confounding effects of arbitrary natural class imbalances.

Varying Dirichlet for Target Label Proportion. To purely isolate and stress-test the effect of label shift intensity, the source marginal distribution p_Y is fixed to a uniform distribution ($p_Y(y) = 1/K$ for all $y \in \mathcal{Y}$). The target marginal distribution q_Y, however, is generated dynamically using a Dirichlet prior $\text{Dir}(10^a \cdot \mathbf{1}_K)$. We set the concentration parameter $a \in \{-3, -2, -1, 0, 1, 2, 3\}$. This explicit parameterization allows us to simulate a comprehensive spectrum of label shift intensities: $a = -3$ yields higher sparsity across class labels, resulting in a more extreme label shift, while $a = 3$ yields a near-uniform distribution representing negligible shift.

Subset Size Configurations. The sample size configurations are tailored to the label dimension of each benchmark dataset to ensure statistical validity. The calibration set sizes n are fixed at $10,000$ for CIFAR-100, $2,000$ for 20 Newsgroups, and 700 for FER2013. The testing set size used for evaluating prediction coverage is consistently set to $1,000$ across all trials. During the estimation sample size settings, we vary the source estimation size n_s, while the target estimation size is strictly maintained at $n_t = 2n_s$.

M STATISTICAL HONESTY RESULTS ON SYNTHETIC DATASETS

M.1 EFFICIENCY WITH CALIBRATION SAMPLE SIZE.

Figure 6 shows the impact of the calibration sample size n on finite-sample variance and algorithmic scalability. The Oracle WCP continues to perfectly anchor the 90.0% nominal target with a constant average set size of 10.88 across all scaling regimes. QuiCK.polytope and QuiCK.hyperbox demonstrate a remarkable *asymptotic tightening* property. At a smaller calibration size ($n = 5,000$), the empirical CDF is highly discrete; to safely hedge against this finite-sample evaluation variance, QuiCK strictly preserves validity by outputting conservative coverage (97.1%–97.4%) and correspondingly wider sets (16.2–16.7). As n scales to $200,000$ and the empirical distribution smoothens, QuiCK safely shrinks its prediction sets (≈ 12.3) and asymptotically converges toward the nominal coverage target ($\approx 92.6\%$). Notably, the polytope formulation consistently yields tighter bounding geometries than the hyperbox across all sample sizes. Because this setting isolates n while fixing the estimation samples at a massive default size of $n_s = n_t = 20,000$ (as detailed in Appendix L.3), Plug-in WCP are near-perfect. BBHR similarly stabilizes at 90.6% coverage with an 11.14 set size as corner samples may not be sufficiently drawn. Notably, all methods show a decreasing trend in statistical uncertainties as n increases.

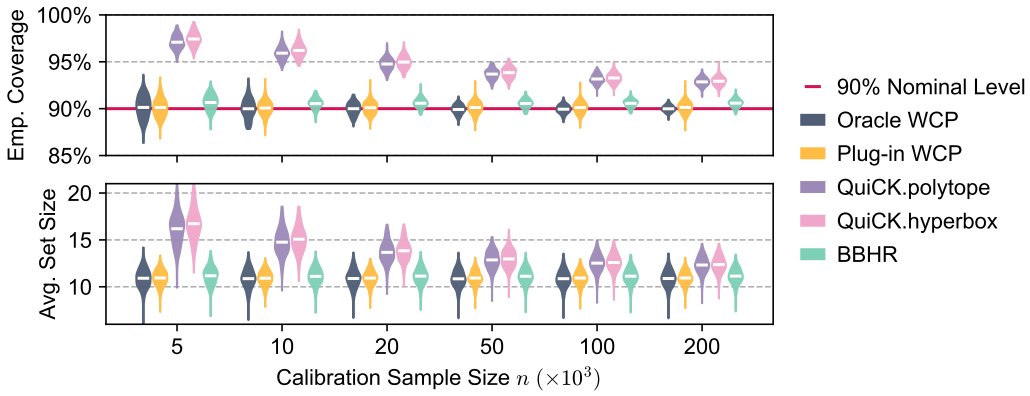


Figure 6: Empirical coverage and average prediction set size on synthetic datasets varying calibration sample size n ($K = 100$, 200 trials). Top: Empirical coverage relative to the 90% nominal level (magenta line). Bottom: Average prediction set size.

M.2 LABEL SPACE SCALING AND UNCERTAINTY PROPAGATION.

Figure 7 exhibits methods' behaviors by scaling the number of classes K from 5 to 200 while maintaining the effective estimation sample size $200 \times K$. The Oracle WCP consistently achieves the exact 90.0% nominal coverage regardless of dimensionality, with average set sizes naturally expanding from 3.19 to 11.85 as the label space grows.

QuiCK.polytope and QuiCK.hyperbox, successfully guarantee strict validity across all K regimes (90.3%–94.0%). Note that QuiCK's results seem more conservative as K grows, but it is a faithful propagation of input statistical uncertainty. In large label spaces, constructing similar length of confidence intervals for $\mathbf{C}^*$ and $\mathbf{q}^*$ inherently needs larger estimation sample sizes. QuiCK exactly propagates this input uncertainty to the final bounds, naturally leading to over-coverage (e.g., 93.4% and 94.0% at $K = 200$). This input-driven nature will be further corroborated in our real-world CIFAR-100 experiments (see Figure 2), where providing more estimation data immediately shrinks the input intervals and tightens QuiCK's outputs. Between the two geometries, QuiCK.polytope operates directly on the exact joint constraint, effectively reducing the dimensional slack to yield slightly tighter sets (13.89) than the element-wise QuiCK.hyperbox (14.34).

Again, because the estimation sample size is proportionally scaled in this setting, it provides massive data at large K (e.g., 40,000 samples at $K = 200$). Therefore, Plug-in WCP remain accurate, but BBHR shows unstable performances as K varies.

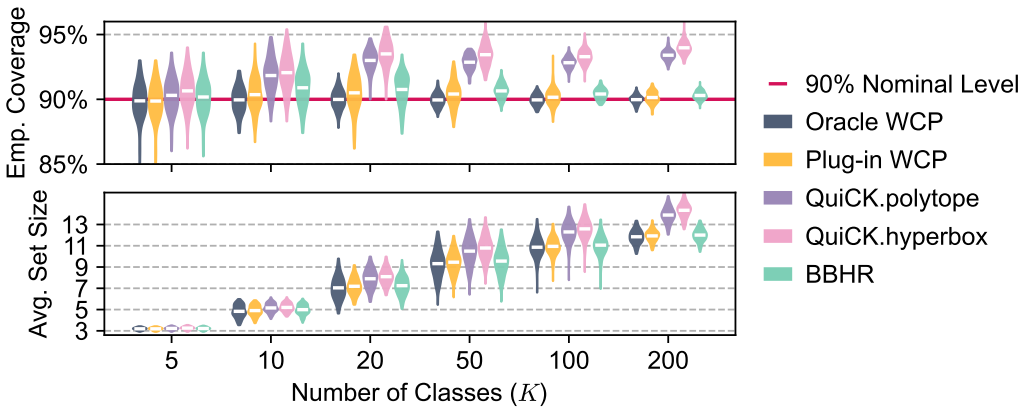


Figure 7: Empirical coverage and average prediction set size on synthetic datasets varying the number of classes K (200 trials). Top: Empirical coverage relative to the 90% nominal level (magenta line). Bottom: Average prediction set size.

M.3 Strictness of Nominal Coverage Setting

We evaluate how the framework responds to increasing stringency in uncertainty quantification by varying the nominal target coverage $1 - \alpha$ from 90.0% to 99.5% (Figure 8). The Oracle WCP consistently matches the exact nominal targets, with average set sizes naturally expanding from 10.86 to 22.82 to fulfill the stricter probabilistic requirements.

QuiCK.polytope and QuiCK.hyperbox successfully maintain strict validity across all target levels. As the nominal coverage requirement increases, the frameworks safely expand their prediction sets to hedge against the propagated estimation uncertainty, ensuring safe over-coverage (e.g., 99.88% and 99.89% at the 99.5% target, respectively). Consistent with previous geometric observations, QuiCK.polytope yields slightly tighter bounding geometries (27.92 at 99.5%) than the element-wise QuiCK.hyperbox (28.28) across all target levels.

Finally, we report the performance of Plug-in WCP and BBHR. Because this specific setting also employs the default massive estimation sample size ($n_s = n_t = 20,000$), the point-estimated weights are highly accurate. Consequently, Plug-in WCP and BBHR manage to closely track the nominal targets without exhibiting their typical under-coverage, producing set sizes comparable to the Oracle.

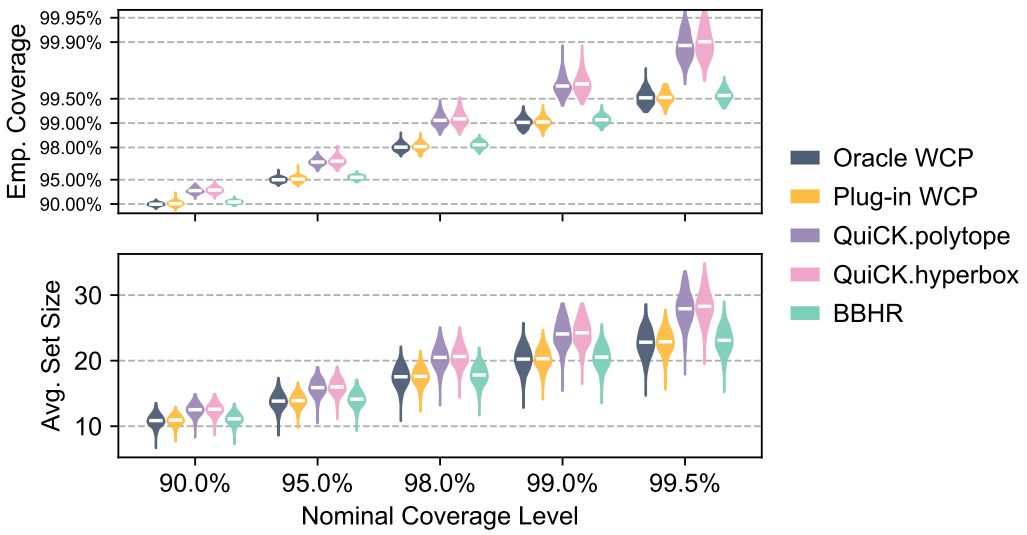


Figure 8: Empirical coverage and average prediction set size on synthetic datasets varying the nominal coverage level $1 - \alpha$ ($K = 100$, 200 trials). Top: Empirical coverage relative to the target nominal levels. Bottom: Average prediction set size.

N STATISTICAL HONESTY RESULTS FOR BENCHMARKS

N.1 ADAPTABILITY TO TARGET TOPOLOGY ON CIFAR-100

We further evaluate the framework's adaptability across different label shift intensities, varying the Dirichlet drift parameter a from -3.0 (extreme concentration) to 3.0 (near-uniform distribution).

We observe that the QuiCK variants performs well under the highly skewed shift ($a = -3.0$) while becoming increasingly conservative as the target distribution approaches uniformity. This is because the n_t estimation samples are concentrated on fewer classes when $a = -3.0$, yielding tight confidence intervals for the "active" classes and producing a relatively compact prediction set size (24.7 for QuiCK.polytope). However, as a increases to 3.0 and the target distribution becomes uniform, the fixed samples are spread thinly across all $K = 100$ classes. This dramatically dilutes the effective sample size per class, widening the Clopper-Pearson confidence intervals for the target proportions. Consequently, this expanded input uncertainty propagates into a wider joint confidence region for the importance weights, compelling QuiCK to conservatively expand its prediction sets (up to 39.7) to strictly preserve *finite-sample validity*. Regardless of the target distribution's skewness, the Oracle WCP establishes a highly stable average set size (≈ 17.3), while Plug-in WCP and BBHR show comparable results to those of Oracle WCP.

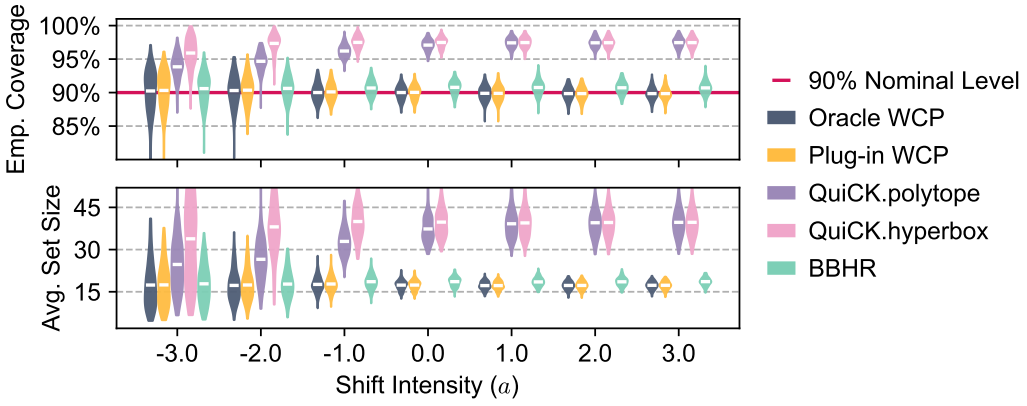


Figure 9: Empirical coverage and average prediction set size on the CIFAR-100 benchmark varying the Dirichlet shift intensity a. The estimation sample size is fixed at an abundant scale ($n_s = 10,000, n_t = 2 \cdot n_s$). Top: Empirical coverage. Bottom: Average prediction set size.

N.2 PREDICTION ON FER2013 AND 20 NEWSGROUPS

In this section, we present the extended real-world benchmark results on the lower-dimensional FER2013 ($K = 7$) and 20 Newsgroups ($K = 20$) datasets. We conduct the same setting study as in the main text: varying the estimation sample size (under a fixed skewed shift $a = -2$) and varying the Dirichlet shift intensity a.

As shown in the experimental data, the behaviors of all methods on these lower-dimensional datasets mirror the theoretical properties observed on CIFAR-100. For both FER2013 and 20 Newsgroups:

- **Asymptotic Tightening:** As the estimation sample size increases to 5,000, the statistical uncertainty in the importance weights significantly diminishes. Consequently, the prediction set sizes of QuiCK smoothly shrink (e.g., from 4.8 to 3.4 on 20 Newsgroups), asymptotically approaching the Oracle WCP lower bound while safely preserving the 90% nominal coverage.
- **Geometric Advantage:** The performance gap between polytope and hyperbox bounding remain pronounced in shift intensity settings. It consistently successfully suppresses the dimensional slack that plagues the element-wise QuiCK.hyperbox.

- **Topological Adaptability:** Under varying shift intensities a, QuiCK continues to strictly guarantee coverage. As a approaches uniformity ($a \to 3.0$), the effective sample size per class is diluted, causing a natural and mathematically justified expansion in the prediction sets (e.g., from 3.4 to 4.1 for QuiCK.polytope on 20 Newsgroups) to hedge against the denser accumulation of estimation uncertainty.

These supplementary results confirm that QuiCK's rigorous *finite-sample guarantees* and geometric efficiencies generalize seamlessly across datasets of varying dimensionalities and complexities.

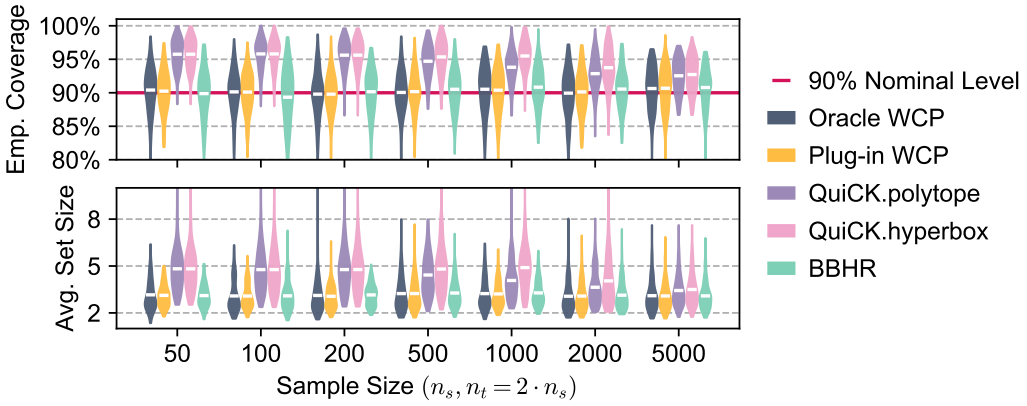


Figure 10: Empirical coverage and average prediction set size on the 20 Newsgroups dataset varying estimation sample sizes n_s and $n_t = 2 \cdot n_s$ under a skewed shift ($a = -2$). QuiCK variants demonstrate smooth asymptotic tightening, naturally shrinking their prediction sets to approach the Oracle lower bound as estimation uncertainty diminishes with larger sample sizes. The geometric advantage of QuiCK.polytope over QuiCK.hyperbox remains prominent in this 20-dimensional space.

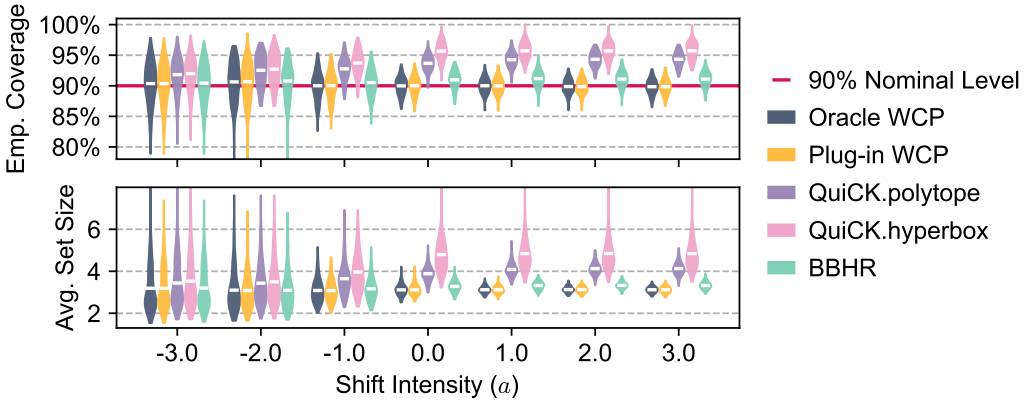


Figure 11: Empirical coverage and average prediction set size on the 20 Newsgroups dataset varying the Dirichlet shift intensity a with a fixed estimation sample size ($n_t = 5,000$). Oracle WCP remains highly stable. The QuiCK frameworks perfectly adapt to the topological shift, expanding prediction sets to hedge against the denser accumulation of estimation uncertainty as the distribution approaches uniformity, thereby strictly guaranteeing validity across all conditions.

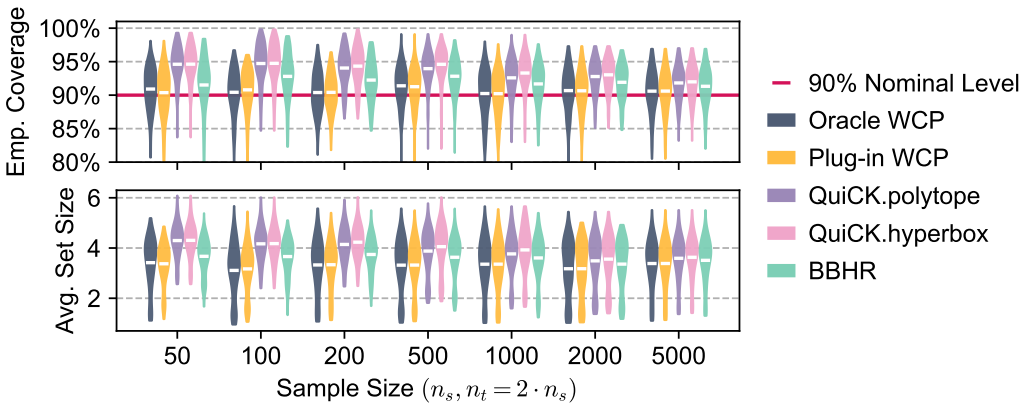


Figure 12: Empirical coverage and average prediction set size on the low-dimensional FER2013 dataset varying estimation sample sizes n_s and $n_t = 2 \cdot n_s$ under a skewed shift ($a = -2$). At $K = 7$, QuiCK variants perfectly provide safe over-coverage during data scarcity and seamlessly converge toward the Oracle limit as the sample size increases. Both bounding geometries perform reliably, with QuiCK.polytope maintaining a slight edge.

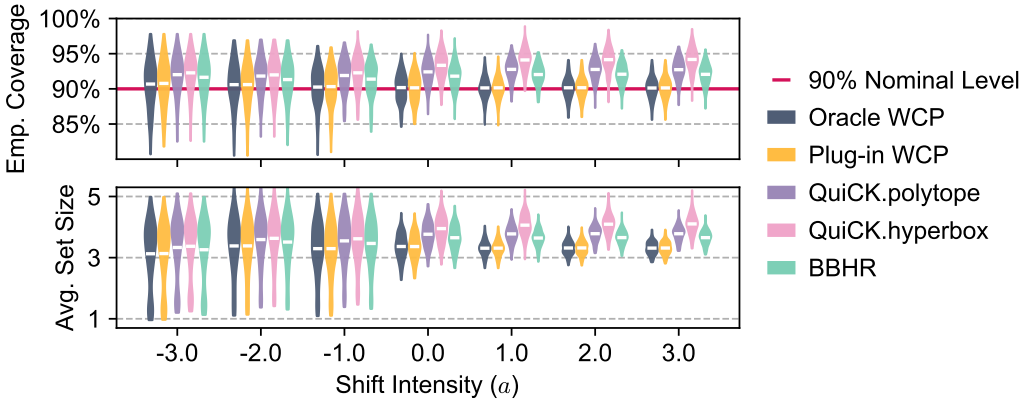


Figure 13: Empirical coverage and average prediction set size on the FER2013 dataset varying the Dirichlet shift intensity a with a fixed estimation sample size ($n_t = 5,000$). The frameworks robustly adapt to the topological changes in the target distribution, dynamically scaling their bounding geometries to rigorously maintain the 90% nominal coverage target.

N.3 Computational Statistics for Synthetic Datasets

Table 2: End-to-end runtime (milliseconds) across 200 trials varying the number of classes K. The "+" notation indicates the inclusion of prerequisite computational overhead: solving for the point estimate $\widehat{\omega}$ (Plug-in WCP+), constructing the hyperbox bounds (QuiCK.hyperbox+), and MCMC sampling within the polytope (BBHR.polytope+).

K	Oracle WCP	Plug-in WCP+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
5	0.006 ± 0.003	0.049 ± 0.024	1.138 ± 0.529	0.488 ± 0.323	3.610 ± 1.227
10	0.006 ± 0.002	0.056 ± 0.009	1.330 ± 0.149	0.636 ± 0.099	4.531 ± 0.747
20	0.008 ± 0.003	0.096 ± 0.028	2.240 ± 0.499	1.477 ± 0.174	8.311 ± 1.575
50	0.013 ± 0.002	0.298 ± 0.079	8.155 ± 0.923	14.935 ± 1.420	27.947 ± 1.271
100	0.032 ± 0.005	0.954 ± 0.034	19.932 ± 2.848	29.995 ± 7.004	185.249 ± 6.322
200	0.108 ± 0.020	4.526 ± 0.470	91.550 ± 8.133	188.293 ± 63.630	412.902 ± 26.279

Table 3: Optimization statistics for QuiCK.polytope varying K (200 trials).

K	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
5	17.710 ± 0.454	54.700 ± 16.184	3.731 ± 2.579
10	17.715 ± 0.451	74.000 ± 10.100	3.754 ± 2.468
20	17.685 ± 0.465	84.425 ± 6.693	4.361 ± 2.618
50	17.710 ± 0.454	89.400 ± 4.686	5.260 ± 2.799
100	17.690 ± 0.462	93.750 ± 2.927	5.951 ± 3.314
200	17.650 ± 0.477	96.040 ± 1.486	6.810 ± 3.590

Table 4: End-to-end runtime (milliseconds) across 200 trials varying calibration sample size n. The "+" notation indicates prerequisite computational overhead.

n ($\times 10^3$)	Oracle	Plug-in WCP+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
5	0.024 ± 0.008	1.163 ± 0.362	18.395 ± 3.959	51.240 ± 15.472	177.632 ± 9.634
10	0.026 ± 0.013	1.294 ± 1.076	18.373 ± 3.297	61.556 ± 17.883	178.400 ± 9.594
20	0.028 ± 0.034	1.106 ± 0.279	17.256 ± 2.359	69.021 ± 17.847	179.286 ± 8.328
50	0.031 ± 0.013	1.111 ± 0.245	18.230 ± 2.037	85.462 ± 18.188	182.605 ± 8.170
100	0.035 ± 0.023	1.301 ± 2.399	20.098 ± 5.630	94.631 ± 16.549	184.929 ± 12.743
200	0.040 ± 0.005	1.064 ± 0.200	19.254 ± 1.897	99.077 ± 11.678	185.202 ± 5.158

Table 5: End-to-end runtime (milliseconds) across 200 trials varying estimation sample sizes n_s, n_t. The "+" notation indicates the inclusion of prerequisite computational overhead: solving for the point estimate $\widehat{\omega}$ (Plug-in WCP+), constructing the hyperbox bounds (QuiCK.hyperbox+), and MCMC sampling within the polytope (BBHR.polytope+).

(n_s, n_t)	Oracle WCP	Plug-in WCP+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
50	0.006 ± 0.002	0.058 ± 0.012	1.124 ± 0.209	0.373 ± 0.843	4.531 ± 0.924
100	0.006 ± 0.001	0.059 ± 0.006	1.193 ± 0.448	0.349 ± 0.118	4.463 ± 1.486
200	0.012 ± 0.064	0.064 ± 0.022	1.219 ± 0.261	0.435 ± 0.330	4.708 ± 1.062
500	0.007 ± 0.002	0.064 ± 0.012	1.259 ± 0.251	0.472 ± 0.254	4.686 ± 0.700
1000	0.006 ± 0.002	0.061 ± 0.017	1.205 ± 0.325	0.426 ± 0.324	4.596 ± 0.659

Table 6: QuiCK.polytope LP running statistics (200 trials) across varying estimation sample sizes n_s, n_t.

(n_s, n_t)	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
50	17.380 ± 0.875	90.000 ± 0.000	4.475 ± 3.119
100	17.640 ± 0.480	90.000 ± 0.000	4.475 ± 3.119
200	17.650 ± 0.477	90.000 ± 0.000	4.475 ± 3.119
500	17.665 ± 0.472	89.800 ± 1.400	4.442 ± 3.110
1000	17.675 ± 0.468	88.600 ± 3.747	4.334 ± 2.972

Table 7: QuiCK.polytope LP running statistics (200 trials) across varying calibration sample size n.

n $(\times 10^3)$	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
5	13.355 ± 0.479	91.515 ± 4.031	5.286 ± 3.051
10	14.395 ± 0.489	94.780 ± 3.144	6.717 ± 3.791
20	15.360 ± 0.480	96.285 ± 2.307	7.747 ± 4.078
50	16.665 ± 0.482	97.130 ± 1.801	8.576 ± 4.555
100	17.680 ± 0.498	97.565 ± 1.551	9.334 ± 4.853
200	18.585 ± 0.550	97.875 ± 1.490	9.489 ± 4.733

Table 8: End-to-end runtime (milliseconds) across 200 trials varying nominal coverage level $1 - \alpha$. The "+" notation indicates prerequisite computational overhead.

Coverage	Oracle	Plug-in WCP+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
0.9	0.033 ± 0.009	1.070 ± 0.229	19.025 ± 2.703	93.704 ± 13.593	183.271 ± 8.179
0.95	0.035 ± 0.007	1.373 ± 1.395	20.537 ± 5.050	95.328 ± 18.568	187.598 ± 17.782
0.98	0.033 ± 0.004	1.124 ± 0.348	21.376 ± 2.697	93.778 ± 14.442	185.790 ± 8.744
0.99	0.027 ± 0.004	1.167 ± 0.497	22.412 ± 3.640	94.695 ± 15.022	179.818 ± 10.399
0.995	0.027 ± 0.022	1.146 ± 0.382	24.871 ± 4.027	94.287 ± 14.314	178.566 ± 11.165

Table 9: QuiCK.polytope LP running statistics (200 trials) across varying nominal coverage level $1 - \alpha$.

Coverage	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
0.9	17.680 ± 0.498	97.565 ± 1.551	9.334 ± 4.853
0.95	17.625 ± 0.504	96.950 ± 2.095	8.857 ± 4.469
0.98	17.600 ± 0.500	96.615 ± 2.125	8.704 ± 3.693
0.99	17.580 ± 0.542	95.620 ± 2.695	7.573 ± 3.532
0.995	17.590 ± 0.531	92.530 ± 3.667	5.772 ± 2.787

N.4 COMPUTATIONAL STATISTICS FOR BENCHMARKS

CIFAR-100 The empirical results highlight the key optimization strengths of QuiCK.polytope. In lower dimensions ($K \leq 20$), QuiCK.hyperbox+ runs slightly faster than QuiCK.polytope due to its uncoupled element-wise formulations. However, on CIFAR-100 ($K = 100$), QuiCK.polytope achieves a remarkable class pruning rate (ranging from 78.1% up to 97.7%). This massive reduction in active optimization variables allows QuiCK.polytope (12–15 ms) to reverse the efficiency gap and consistently outperform QuiCK.hyperbox+ (15–22 ms).

Across all datasets, the number of base Linear Programming evaluations (`Base LP Calls`) remains strictly invariant to both sample scarcity and label shift intensity. For instance, base LP calls remain anchored at ≈ 13.3 for CIFAR-100, ≈ 11.0 for 20 Newsgroups, and ≈ 10.5 for FER2013 across all conditions. This confirms that the computational complexity of QuiCK is majorly influenced by the underlying geometric pruning search rather than the amount of the calibration data.

Table 10: End-to-end runtime (milliseconds) on the CIFAR-100 benchmark across 200 runs varying estimation sample size n_s (with $n_t = 2 \cdot n_s$ and $a = -2.0$). The "+" notation indicates the inclusion of prerequisite computational overhead.

n_s	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
200	0.022 ± 0.003	0.915 ± 0.465	10.629 ± 2.790	9.485 ± 0.949	171.562 ± 4.734
500	0.022 ± 0.002	1.051 ± 0.104	9.795 ± 2.314	9.993 ± 0.627	173.091 ± 4.207
1000	0.026 ± 0.010	1.141 ± 0.250	9.858 ± 1.715	11.491 ± 0.896	207.221 ± 47.330
2000	0.027 ± 0.011	1.155 ± 0.204	11.047 ± 2.028	13.129 ± 1.467	195.825 ± 39.189
5000	0.027 ± 0.011	1.160 ± 0.208	12.177 ± 2.241	15.635 ± 2.417	192.985 ± 30.071
10000	0.027 ± 0.011	1.180 ± 0.392	15.543 ± 2.759	20.903 ± 5.687	182.624 ± 13.821

Table 11: QuiCK.polytope optimization statistics on the CIFAR-100 benchmark varying estimation sample size n_s (200 trials).

n_s	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
200	13.350 ± 0.477	97.915 ± 1.303	8.532 ± 3.815
500	13.350 ± 0.477	97.915 ± 1.303	8.532 ± 3.815
1000	13.350 ± 0.477	97.890 ± 1.345	8.455 ± 3.834
2000	13.325 ± 0.468	96.895 ± 1.893	7.134 ± 3.496
5000	13.300 ± 0.469	88.000 ± 6.180	4.171 ± 2.634
10000	13.345 ± 0.516	78.120 ± 11.298	3.516 ± 2.073

Table 12: End-to-end runtime (milliseconds) on the CIFAR-100 benchmark across 200 runs varying the Dirichlet shift intensity a (with $n_t = 20,000$). The "+" notation indicates prerequisite computational overhead.

Shift Inst. a	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
-3.0	0.027 ± 0.011	1.141 ± 0.275	15.175 ± 3.329	22.432 ± 5.787	183.897 ± 14.480
-2.0	0.027 ± 0.011	1.180 ± 0.392	15.543 ± 2.759	20.903 ± 5.687	182.624 ± 13.821
-1.0	0.029 ± 0.013	1.331 ± 2.329	14.078 ± 2.161	16.006 ± 2.339	185.676 ± 20.037
0.0	0.031 ± 0.023	1.259 ± 0.502	12.891 ± 2.913	14.214 ± 2.008	186.747 ± 18.017
1.0	0.028 ± 0.012	1.228 ± 0.290	12.826 ± 2.471	14.155 ± 2.081	187.979 ± 16.099
2.0	0.028 ± 0.011	1.249 ± 0.374	13.084 ± 2.381	15.572 ± 8.703	189.108 ± 15.801
3.0	0.028 ± 0.012	1.323 ± 0.601	13.287 ± 3.048	14.762 ± 5.751	190.740 ± 20.801

Table 13: QuiCK.polytope optimization statistics on the CIFAR-100 benchmark varying the Dirichlet shift intensity a (200 trials).

Shift Inst. a	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
-3.0	13.330 ± 0.521	72.100 ± 15.647	2.979 ± 1.873
-2.0	13.345 ± 0.516	78.120 ± 11.298	3.516 ± 2.073
-1.0	13.465 ± 0.518	89.665 ± 4.215	4.155 ± 2.656
0.0	13.235 ± 0.458	95.245 ± 2.228	6.050 ± 3.257
1.0	13.325 ± 0.489	97.030 ± 1.775	7.203 ± 3.885
2.0	13.345 ± 0.486	97.600 ± 1.442	7.911 ± 3.895
3.0	13.325 ± 0.489	97.735 ± 1.387	8.053 ± 3.839

20 Newsgroups (K=20) Below shows results for 20 Newsgroups.

Table 14: End-to-end runtime (milliseconds) on the 20 Newsgroups benchmark across 200 runs varying estimation sample size n_s (with $n_t = 2 \cdot n_s$ and $a = -2.0$).

n_s	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
50	0.004 ± 0.001	0.078 ± 0.014	1.191 ± 0.155	0.472 ± 0.068	6.751 ± 0.204
100	0.004 ± 0.001	0.093 ± 0.012	1.277 ± 0.166	0.534 ± 0.088	6.845 ± 0.255
200	0.004 ± 0.001	0.092 ± 0.010	1.323 ± 0.143	0.573 ± 0.228	7.018 ± 0.216
500	0.004 ± 0.001	0.092 ± 0.009	1.353 ± 0.191	0.685 ± 0.105	7.014 ± 0.195
1000	0.004 ± 0.001	0.092 ± 0.009	1.377 ± 0.182	0.741 ± 0.112	7.044 ± 0.221
2000	0.005 ± 0.001	0.095 ± 0.010	1.500 ± 0.220	0.858 ± 0.373	7.032 ± 0.216
5000	0.004 ± 0.001	0.094 ± 0.008	1.526 ± 0.184	0.909 ± 0.131	7.020 ± 0.197

Table 15: QuiCK.polytope optimization statistics on the 20 Newsgroups benchmark varying estimation sample size n_s (200 trials).

n_s	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
50	10.985 ± 0.212	90.325 ± 5.125	6.162 ± 3.140
100	10.955 ± 0.365	90.375 ± 5.314	6.228 ± 3.187
200	10.990 ± 0.099	90.350 ± 5.160	6.164 ± 3.143
500	10.975 ± 0.156	80.750 ± 9.107	4.434 ± 2.765
1000	10.980 ± 0.140	68.075 ± 14.737	3.472 ± 1.881
2000	10.975 ± 0.156	57.850 ± 17.990	2.924 ± 1.540
5000	10.980 ± 0.140	53.875 ± 22.161	2.674 ± 1.578

Table 16: End-to-end runtime (milliseconds) on the 20 Newsgroups benchmark across 200 runs varying the Dirichlet shift intensity a (with $n_t = 5,000$).

Shift Inst. a	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
-3.0	0.004 ± 0.001	0.093 ± 0.007	1.525 ± 0.211	0.862 ± 0.128	6.973 ± 0.156
-2.0	0.004 ± 0.001	0.094 ± 0.008	1.526 ± 0.184	0.909 ± 0.131	7.020 ± 0.197
-1.0	0.005 ± 0.001	0.100 ± 0.012	1.668 ± 0.256	1.121 ± 0.293	7.196 ± 0.253
0.0	0.005 ± 0.001	0.097 ± 0.012	1.711 ± 0.532	1.064 ± 0.533	7.229 ± 0.274
1.0	0.004 ± 0.001	0.093 ± 0.008	1.743 ± 0.241	0.928 ± 0.097	7.142 ± 0.158
2.0	0.004 ± 0.001	0.095 ± 0.009	1.841 ± 0.271	0.946 ± 0.113	7.169 ± 0.154
3.0	0.004 ± 0.001	0.092 ± 0.007	1.846 ± 0.351	0.928 ± 0.141	7.153 ± 0.164

Table 17: QuiCK.polytope optimization statistics on the 20 Newsgroups benchmark varying the Dirichlet shift intensity a (200 trials).

Shift Inst. a	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
-3.0	10.980 ± 0.140	52.550 ± 21.319	2.592 ± 1.641
-2.0	10.980 ± 0.140	53.875 ± 22.161	2.674 ± 1.578
-1.0	10.990 ± 0.099	54.100 ± 16.589	2.805 ± 1.561
0.0	10.965 ± 0.184	54.300 ± 10.619	2.908 ± 1.572
1.0	10.965 ± 0.184	54.675 ± 10.381	2.607 ± 1.486
2.0	10.970 ± 0.171	53.150 ± 10.912	2.880 ± 1.487
3.0	10.965 ± 0.184	52.950 ± 10.679	2.844 ± 1.622

FER2013 (K=7) Below shows results for FER2013.

Table 18: End-to-end runtime (milliseconds) on the FER2013 benchmark across 200 runs varying estimation sample size n_s (with $n_t = 2 \cdot n_s$ and $a = -2.0$).

n_s	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
50	0.002 ± 0.000	0.044 ± 0.006	0.677 ± 0.066	0.235 ± 0.040	3.274 ± 0.104
100	0.002 ± 0.000	0.042 ± 0.004	0.648 ± 0.071	0.223 ± 0.020	3.248 ± 0.115
200	0.003 ± 0.002	0.046 ± 0.012	0.648 ± 0.089	0.271 ± 0.083	3.332 ± 0.291
500	0.003 ± 0.001	0.047 ± 0.008	0.605 ± 0.081	0.265 ± 0.032	3.342 ± 0.133
1000	0.003 ± 0.002	0.046 ± 0.011	0.616 ± 0.116	0.282 ± 0.067	3.316 ± 0.145
2000	0.003 ± 0.001	0.049 ± 0.011	0.620 ± 0.104	0.280 ± 0.045	3.328 ± 0.143
5000	0.003 ± 0.001	0.046 ± 0.006	0.601 ± 0.092	0.272 ± 0.067	3.285 ± 0.101

Table 19: QuiCK.polytope optimization statistics on the FER2013 benchmark varying estimation sample size n_s (200 trials).

n_s	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
50	10.520 ± 0.500	79.071 ± 9.676	4.248 ± 2.539
100	10.515 ± 0.500	78.357 ± 9.895	4.172 ± 2.530
200	10.510 ± 0.500	70.857 ± 13.690	3.552 ± 2.186
500	10.560 ± 0.496	56.357 ± 22.140	2.970 ± 1.729
1000	10.550 ± 0.497	50.143 ± 23.776	2.925 ± 1.689
2000	10.525 ± 0.499	44.643 ± 25.271	2.964 ± 1.560
5000	10.510 ± 0.500	47.786 ± 23.500	2.951 ± 1.595

Table 20: End-to-end runtime (milliseconds) on the FER2013 benchmark across 200 runs varying the Dirichlet shift intensity a (with $n_t = 5,000$).

Shift Inst. a	Oracle	Point Est.+	QuiCK.polytope	QuiCK.hyperbox+	BBHR.polytope+
-3.0	0.002 ± 0.000	0.042 ± 0.005	0.574 ± 0.079	0.249 ± 0.026	3.218 ± 0.071
-2.0	0.003 ± 0.001	0.046 ± 0.006	0.601 ± 0.092	0.272 ± 0.067	3.285 ± 0.101
-1.0	0.003 ± 0.001	0.049 ± 0.011	0.630 ± 0.187	0.312 ± 0.065	3.342 ± 0.230
0.0	0.003 ± 0.001	0.046 ± 0.006	0.591 ± 0.074	0.317 ± 0.032	3.339 ± 0.092
1.0	0.003 ± 0.001	0.055 ± 0.012	0.616 ± 0.076	0.348 ± 0.036	3.436 ± 0.142
2.0	0.003 ± 0.001	0.054 ± 0.011	0.631 ± 0.089	0.357 ± 0.040	3.421 ± 0.137
3.0	0.003 ± 0.001	0.050 ± 0.012	0.611 ± 0.171	0.345 ± 0.083	3.368 ± 0.164

Table 21: QuiCK.polytope optimization statistics on the FER2013 benchmark varying the Dirichlet shift intensity a (200 trials).

Shift Inst. a	Base LP Calls	Pruned Class (%)	Avg. Calls/ Unpruned Class
-3.0	10.510 ± 0.500	44.429 ± 26.411	2.518 ± 1.632
-2.0	10.510 ± 0.500	47.786 ± 23.500	2.951 ± 1.595
-1.0	10.520 ± 0.500	39.357 ± 21.829	2.624 ± 1.370
0.0	10.545 ± 0.498	26.143 ± 15.193	2.203 ± 0.949
1.0	10.545 ± 0.498	16.500 ± 10.010	2.004 ± 0.864
2.0	10.570 ± 0.495	15.929 ± 9.916	2.096 ± 0.848
3.0	10.555 ± 0.497	15.571 ± 10.019	1.991 ± 0.836