
Building a stable classifier with the inflated argmax

Jake A. Soloff

Department of Statistics
University of Chicago
Chicago, IL 60637
soloff@uchicago.edu

Rina Foygel Barber

Department of Statistics
University of Chicago
Chicago, IL 60637
rina@uchicago.edu

Rebecca Willett

Departments of Statistics and Computer Science
NSF-Simons National Institute for Theory and Mathematics in Biology
University of Chicago
Chicago, IL 60637
willett@uchicago.edu

Abstract

We propose a new framework for algorithmic stability in the context of multiclass classification. In practice, classification algorithms often operate by first assigning a continuous score (for instance, an estimated probability) to each possible label, then taking the maximizer—i.e., selecting the class that has the highest score. A drawback of this type of approach is that it is inherently unstable, meaning that it is very sensitive to slight perturbations of the training data, since taking the maximizer is discontinuous. Motivated by this challenge, we propose a pipeline for constructing stable classifiers from data, using bagging (i.e., resampling and averaging) to produce stable continuous scores, and then using a stable relaxation of argmax, which we call the “inflated argmax”, to convert these scores to a set of candidate labels. The resulting stability guarantee places no distributional assumptions on the data, does not depend on the number of classes or dimensionality of the covariates, and holds for any base classifier. Using a common benchmark data set, we demonstrate that the inflated argmax provides necessary protection against unstable classifiers, without loss of accuracy.

1 Introduction

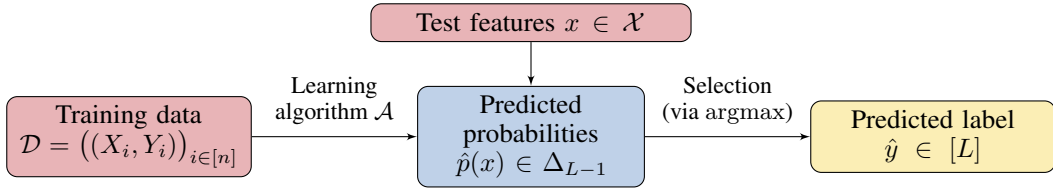
An algorithm that learns from data is considered to be *stable* if small perturbations of the training data do not lead to large changes in its output. Algorithmic stability is an important consideration in many statistical applications. Within the fairness literature, for instance, stability is one aspect of reliable decision-making systems [FSVC19; HV19]. In interpretable machine learning, it similarly serves as a form of reproducibility [MSKA19; YK20; YB24]. [CMX11] relates stability to robustness, where a machine learning algorithm is robust if two samples with similar feature vectors have similar test error. In the context of generalization bounds, [BE02] and subsequent authors study stability of an algorithm’s real-valued output—for instance, values of a regression function. In the setting of a multiclass classification problem, where the data consists of features $X \in \mathcal{X}$ and labels $Y \in [L] = \{1, \dots, L\}$, the results of this literature can thus be applied to analyzing a classifier’s predicted probabilities $\hat{p}_\ell(X) \in [0, 1]$ —i.e., our learned estimates of the conditional probabilities $\Pr\{Y = \ell \mid X\}$, for each $\ell \in [L]$. However, as we will see, stability of these predicted probabilities by no means implies stability of the predicted label itself, $\hat{y} = \operatorname{argmax}_{\ell \in [L]} \hat{p}_\ell(x)$ —an arbitrarily

small perturbation in $\hat{p}_\ell(x)$ might completely change the predicted label \hat{y} . The distinction matters: for system trustworthiness, we care about the model’s final output, on which users base their decisions.

In this paper, we propose a new framework for algorithmic stability in the context of multiclass classification, to define—and achieve—a meaningful notion of stability when the output of the algorithm consists of predicted labels, rather than predicted probabilities. Our work connects to other approaches to allowing for ambiguous classification, including set-valued classification [Gry93; DDB09; SLW19; CDHL21] and conformal inference [PPVG02; VGS05; Lei14; AB23] (we will discuss related work further, below).

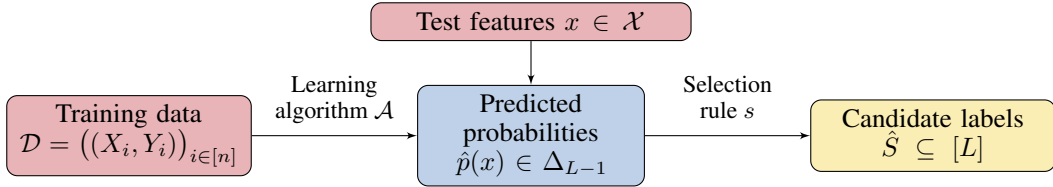
1.1 Problem setting

In supervised classification, we take a data set $\mathcal{D} = ((X_i, Y_i))_{i \in [n]}$ consisting of n observations, and train a classifier that maps from the feature space \mathcal{X} to the set $[L]$ of possible labels.¹ Typically, this map is constructed in two stages. First we run some form of regression to learn a map $\hat{p} : \mathcal{X} \rightarrow \Delta_{L-1}$ from features to predicted probabilities, with $\hat{p}_\ell(x)$ denoting our learned estimate of the conditional label probability, $\Pr\{Y = \ell \mid X = x\}$ (here $\Delta_{L-1} = \{w \in \mathbb{R}^L : w_i \geq 0, \sum_i w_i = 1\}$ denotes the probability simplex in L dimensions). We will write $\hat{p} = \mathcal{A}(\mathcal{D})$, where \mathcal{A} denotes the learning algorithm mapping a data set \mathcal{D} (of any size) to the map \hat{p} . Then the second step is to convert the predicted probabilities $\hat{p}_\ell(x)$ to a predicted label, which is typically done by taking the argmax, i.e., $\hat{y} = \operatorname{argmax}_\ell \hat{p}_\ell(x)$ (with some mechanism for breaking ties). This two-stage procedure can be represented in the following diagram:



When predictions are ambiguous, meaning two or more classes nearly achieve the maximum predicted probability, the selected label becomes unstable and can change based on seemingly inconsequential perturbations to the training data. In other words, the above workflow is fundamentally incompatible with the goal of algorithmic stability. Consequently, in this paper we instead work with *set-valued classifiers*, which return a set of candidate labels—in practice, this typically leads to a singleton set for examples where we have high confidence in the label, but allows for a larger set in the case of ambiguous examples. While the idea of returning a set of candidate labels is not itself new, we will see that the novelty of our work lies in finding a construction that offers provable stability guarantees.

In this framework, a feature vector $x \in \mathcal{X}$ is now mapped to a set of candidate labels $\hat{S} \subseteq [L]$ (rather than a single label \hat{y}), via a selection rule s , as illustrated here:



Formally, given a test point $x \in \mathcal{X}$, this two-stage procedure returns $\hat{S} = s(\hat{p}(x)) \subseteq [L]$, where $\hat{p} = \mathcal{A}(\mathcal{D})$ is the output of the regression algorithm \mathcal{A} trained on data \mathcal{D} . Here $s : \Delta_{L-1} \rightarrow \mathcal{P}([L])$ denotes a *selection rule*, mapping a vector of estimated probabilities to the set of candidate labels, and $\mathcal{P}([L])$ denotes the set of subsets of $[L]$. Of course, the earlier setting—where the procedure returns a single label $\hat{y} = \operatorname{argmax}_\ell \hat{p}_\ell(x)$, rather than a subset—can be viewed as a special case by simply taking s to be the argmax operator.

¹We remark that all the results of this paper apply also to the case of countably infinitely many labels, $L = \infty$, in which case we should take \mathbb{N} to be the label space instead of $[L]$.

If we instead allow for \hat{S} to contain multiple candidate labels, a commonly used version of this framework is given by a top-5 procedure (or top- k , for any fixed k). That is, after running a learning algorithm \mathcal{A} to estimate probabilities $\hat{p}(x)$, the selection rule s then returns the top 5 labels—the labels $\ell_1, \dots, \ell_5 \in [L]$ corresponding to the highest 5 values of $\hat{p}_\ell(x)$. This approach is more stable than a standard argmax. However, the choice of 5 in this setting is somewhat arbitrary, and the set \hat{S} always contains 5 candidate labels, regardless of the difficulty of the test sample—while intuitively, we might expect that it should be possible to return a smaller \hat{S} for test points x that are easier to classify (and a larger \hat{S} if x is more ambiguous). In contrast, in our work we seek a more principled approach, where we provide provable stability guarantees while also aiming for the set \hat{S} to be as small as possible.

1.2 Overview of main results

In this work, we introduce *selection stability*, a new definition of algorithmic stability in the context of classification, which focuses on the stability of the predicted label. We reduce the problem of stabilizing a classifier to separately stabilizing the learning and selection stages, described above. For the selection rule s of the two-stage procedure, we propose the *inflated argmax* operation:

Definition 1 (Inflated argmax). *For any $w \in \mathbb{R}^L$, define the inflated argmax as*

$$\operatorname{argmax}^\varepsilon(w) := \{j \in [L] : \operatorname{dist}(w, R_j^\varepsilon) < \varepsilon\}, \quad (1)$$

where $\operatorname{dist}(w, R_j^\varepsilon) = \inf_{v \in R_j^\varepsilon} \|w - v\|$, and where

$$R_j^\varepsilon = \left\{ w \in \mathbb{R}^L : w_j \geq \max_{\ell \neq j} w_\ell + \varepsilon/\sqrt{2} \right\}.$$

Our procedure will then return the set $\hat{S} = \operatorname{argmax}^\varepsilon(\hat{p}(x))$ of candidate labels—intuitively, $j \in \operatorname{argmax}^\varepsilon(\hat{p}(x))$ indicates that a small perturbation of the predicted probabilities, $\hat{p}(x)$, would lead to the j th label’s predicted probability being largest by some margin. In particular, by construction, the inflated argmax will always include any maximal index—that is, if $\hat{p}(x)_j$ is the (possibly non-unique) largest estimated probability, then we must have $j \in \hat{S} = \operatorname{argmax}^\varepsilon(\hat{p}(x))$ (this fact, and other properties of the inflated argmax, will be established formally in Proposition 10 below).

In this work, we derive the stabilizing properties of the inflated argmax, and give an algorithm to compute it efficiently. We prove that combining this operation with bagging at the learning step will provably stabilize any classifier. In particular, our guarantee holds with no assumptions on the data, and no constraints on the dimensionality of the covariates nor on the number of classes.

2 Framework: stable classification

In this section, we propose a definition of algorithmic stability in the setting of multiclass classification. To begin, we formally define a *classification algorithm* as a map²

$$\mathcal{C} : \cup_{n \geq 0} (\mathcal{X} \times [L])^n \times \mathcal{X} \longrightarrow \mathcal{O}([L]),$$

which maps a training data set \mathcal{D} of any size n , together with a test feature vector $x \in \mathcal{X}$, to a candidate set of labels $\hat{S} = \mathcal{C}(\mathcal{D}, x)$. To relate this notation to our earlier terminology, the two-stage selection procedure described in Section 1.1 corresponds to the classification algorithm

$$\mathcal{C}(\mathcal{D}, x) = s(\hat{p}(x)) \text{ where } \hat{p} = \mathcal{A}(\mathcal{D}).$$

Abusing notation, we will write this as $\mathcal{C} = s \circ \mathcal{A}$, indicating that \mathcal{C} is obtained by applying the selection rule s to the output of the learning algorithm \mathcal{A} .

We now present our definition of algorithmic stability for a classification algorithm \mathcal{C} . As is common in the algorithmic stability literature, we focus on the stability of the algorithm’s output with respect

²We will assume without comment that \mathcal{C} is measurable. Furthermore, in practice classification algorithms often incorporate randomization (either in the learning stage, such as via stochastic gradient descent, and/or in the selection stage, such as by using a random tie-breaking rule for argmax). All of our definitions and results in this paper can be naturally extended to randomized algorithms—see Appendix B.

to randomly dropping a data point: does the output of \mathcal{C} on a test point x change substantially if we drop a single data point from the training data \mathcal{D} ? If the algorithm’s output were a real-valued prediction $\hat{y} \in \mathbb{R}$, we could assess this by measuring the real-valued change in \hat{y} when a single data point is dropped. For classification, however, we will need to take a different approach:

Definition 2 (Selection stability). *We say a classification algorithm \mathcal{C} has selection stability δ at sample size n if, for all datasets $\mathcal{D} \in (\mathcal{X} \times [L])^n$ and all test features $x \in \mathcal{X}$,*

$$\frac{1}{n} \sum_{i=1}^n \mathbf{1} \left\{ \hat{S} \cap \hat{S}^{\setminus i} = \emptyset \right\} \leq \delta,$$

where $\hat{S} = \mathcal{C}(\mathcal{D}, x)$ and where $\hat{S}^{\setminus i} = \mathcal{C}(\mathcal{D}^{\setminus i}, x)$, for each $i \in [n]$.

Here the notation $\mathcal{D}^{\setminus i}$ denotes the data set \mathcal{D} with i th data point removed—that is, for a data set $\mathcal{D} = ((X_j, Y_j))_{j \in [n]}$, the leave-one-out data set is given by $\mathcal{D}^{\setminus i} = ((X_j, Y_j))_{j \in [n] \setminus i}$.

2.1 Interpreting the definition

Intuitively, selection stability controls the probability that our algorithm makes wholly inconsistent claims when dropping a single data point at random from the training set. If we interpret \hat{S} as making the claim that the true label Y is equal to one of the labels in the candidate set \hat{S} , and similarly for $\hat{S}^{\setminus i}$, then as soon as there is *even one single value* that lies in both \hat{S} and $\hat{S}^{\setminus i}$, this means that the two claims are not contradictory—even if the sets are large and are mostly nonoverlapping.

At first glance, this condition appears to be too mild, in the sense that we require the two prediction sets only to have *some* way of being compatible, and allows for substantial difference between the sets \hat{S} and $\hat{S}^{\setminus i}$. However, since standard classification algorithms always output a single label, they often cannot be said to be stable even in this basic sense. Thus, we can view this definition as providing a minimal notion of stability that we should require any interpretable method to satisfy.

2.2 Connection to classical algorithmic stability

Most prior work on algorithmic stability concerns algorithms with continuous outputs—for example, in our notation above, the algorithm \mathcal{A} that returns estimated probabilities $\hat{p}(x)$. With such algorithms, there are more standard tools at our disposal for quantifying and ensuring stability. In this section, we connect classical algorithmic stability to our notion of selection stability (Definition 2). We first recall the following definition.

Definition 3 (Tail stability). [SBW24b] *A learning algorithm \mathcal{A} has tail stability (ε, δ) at sample size n if, for all datasets \mathcal{D} of size n and all test features $x \in \mathcal{X}$,*

$$\frac{1}{n} \sum_{i=1}^n \mathbf{1} \left\{ \|\hat{p}(x) - \hat{p}^{\setminus i}(x)\| \geq \varepsilon \right\} \leq \delta,$$

where $\hat{p} = \mathcal{A}(\mathcal{D})$ and $\hat{p}^{\setminus i} = \mathcal{A}(\mathcal{D}^{\setminus i})$, and where $\|\cdot\|$ denotes the usual Euclidean norm on \mathbb{R}^L .

This intuitive notion of stability is achieved by many well-known algorithms \mathcal{A} , such as nearest-neighbor type methods or methods based on bagging or ensembling (as established by [SBW24a; SBW24b]—see Section 3.1 below for details).

2.3 From stability to selection stability

To construct a classification algorithm using the two-stage procedure outlined in Section 1.1, we need to apply a selection rule s to the output of our learning algorithm \mathcal{A} . We might expect that choosing a stable \mathcal{A} will lead to selection stability in the resulting classification algorithm—but in fact, this is not the case: even if the learning algorithm \mathcal{A} is itself extremely stable in the sense of Definition 3, the classification rule obtained by applying argmax to the output of \mathcal{A} can still be extremely unstable. The underlying issue is that argmax is extremely discontinuous—the perturbation $\|\hat{p}(x) - \hat{p}^{\setminus i}(x)\|$ in the predicted probabilities can be arbitrarily small but still lead to different predicted labels, i.e., $\operatorname{argmax}_{\ell} \hat{p}_{\ell}(x) \neq \operatorname{argmax}_{\ell} \hat{p}_{\ell}^{\setminus i}(x)$.

Since combining argmax with a stable learning algorithm \mathcal{A} will not suffice, we instead seek a different selection rule s —one that will ensure selection stability (when paired with a stable learning algorithm \mathcal{A}). To formalize this aim, we introduce another definition:

Definition 4 (ε -compatibility). *A selection rule $s : \Delta_{L-1} \rightarrow \mathcal{P}([L])$ is ε -compatible if, for any $v, w \in \Delta_{L-1}$,*

$$\|v - w\| < \varepsilon \implies s(v) \cap s(w) \neq \emptyset.$$

This notion of ε -compatibility allows us to construct classification algorithms with selection stability. Combining the above definitions leads immediately to the following result:

Proposition 5. *Let \mathcal{A} be a learning algorithm with tail stability (ε, δ) at sample size n , and let s be a selection rule satisfying ε -compatibility. Then the classification algorithm $\mathcal{C} = s \circ \mathcal{A}$ has selection stability δ at sample size n .*

Therefore, by pairing a stable learning algorithm \mathcal{A} with a compatible selection rule s , we will automatically ensure selection stability of the resulting classification algorithm $\mathcal{C} = s \circ \mathcal{A}$.

Of course, ε -compatibility of the selection rule s might be achieved in a trivial way—for instance, s returns the entire set $[L]$ for any input. As is common in statistical settings (e.g., a tradeoff between Type I error and power, in hypothesis testing problems), our goal is to ensure selection stability while returning an output \hat{S} that is as informative as possible. In particular, later we will consider the specific aim of constructing \hat{S} to be a singleton set as often as possible.

3 Methodology: assumption-free stable classification

In this section, we formally define our pipeline for building a stable classification procedure using any base learning algorithm \mathcal{A} . At a high level, we leverage Proposition 5 and separately address the learning and selection stages described in Section 2.

1. In Section 3.1, we construct a bagged (i.e., ensembled) version of the base learning algorithm \mathcal{A} . The recent work of [SBW24b] ensures that the resulting bagged algorithm has tail stability (ε, δ) , with $\varepsilon \asymp \frac{1}{\sqrt{n\delta}}$.
2. In Section 3.2, we introduce a new selection rule, the inflated argmax , and establish its ε -compatibility. Combined with the bagged algorithm, then, the resulting classification algorithm will be guaranteed to have selection stability δ (by Proposition 5).

Before describing these two steps in detail, we first present our main theorem that characterizes the selection stability guarantee of the resulting procedure. Given sample size n for the training data set \mathcal{D} , the notation $\tilde{\mathcal{A}}_m$ will denote a bagged version of the base algorithm \mathcal{A} , obtained by averaging over subsamples of \mathcal{D} comprised of m data points sampled either with replacement (“bootstrapping”, commonly run with $m = n$) or without replacement (“subbagging”, commonly run with $m = n/2$)—see below for details.

Theorem 6. *Fix any sample size n , any bag size m , and any inflation parameter $\varepsilon > 0$. For any base learning algorithm \mathcal{A} , the classification algorithm $\mathcal{C} = \text{argmax}^\varepsilon \circ \tilde{\mathcal{A}}_m$, obtained by combining the bagged version of \mathcal{A} together with the inflated argmax , satisfies selection stability δ where*

$$\delta = \varepsilon^{-2} \cdot \frac{1 - 1/L}{n - 1} \cdot \frac{p_{n,m}}{1 - p_{n,m}}, \quad (2)$$

where $p_{n,m} = 1 - (1 - \frac{1}{n})^m$ for bootstrapping, and $p_{n,m} = \frac{m}{n}$ for subbagging.

The guarantee in Theorem 6 holds for any base learning algorithm \mathcal{A} , and applies regardless of the complexity of the feature space \mathcal{X} , and allows the test feature $x \in \mathcal{X}$ to be chosen adversarially. The dependence on the number of classes L is mild—in fact, the tail stability parameter δ in (2) differs only by a factor of two for $L = 2$ versus $L = \infty$. Of course, the guarantee does depend on the choice of the bag size m . In general, a smaller value of m leads to a stronger stability guarantee (since $p_{n,m}$ increases with m), but this comes at the expense of accuracy since we are training the base algorithm \mathcal{A} on subsampled data sets \mathcal{D}^r of size m (rather than n). For the common choices of $m = n$ for bootstrap or $m = n/2$ for subbagging, we have $\frac{p_{n,m}}{1 - p_{n,m}} = \mathcal{O}(1)$ for each.

3.1 Bagging classifier weights

In this section, we formally define the construction of a bagged algorithm to recap the tail stability result that our framework leverages. We consider the two most common variants of this ensembling method: bagging (based on bootstrapping training points) and subbagging (based on subsampling). For any data set $\mathcal{D} \in (\mathcal{X} \times [L])^n$, we can define a subsampled data set of size m as follows: for a sequence $r = (i_1, \dots, i_m) \in [n]^m$ (which is called a *bag*), we define the corresponding subsampled data set $\mathcal{D}^r = ((X_i, Y_i))_{i \in r} \in (\mathcal{X} \times [L])^m$. Note that if the bag r contains repeated indices (i.e., $i_k = i_\ell$ for some $k \neq \ell$), then the same data point from the original data set \mathcal{D} will appear multiple times in \mathcal{D}^r .

Definition 7 (Bootstrapping or subbagging a base learning algorithm \mathcal{A}). *Given a data set $\mathcal{D} \in (\mathcal{X} \times [L])^n$ and some $x \in \mathcal{X}$, return the output $\tilde{\mathcal{A}}_m(\mathcal{D})(x) = \mathbb{E}_r[\mathcal{A}(\mathcal{D}^r)(x)]$, where the expected value is taken with respect to a bag r that is sampled as follows:*

- *Bootstrapping (sometimes simply referred to as bagging) [Bre96a; Bre96b] constructs each bag r by sampling m indices $r = (i_1, \dots, i_m)$ uniformly with replacement from $[n]$.*
- *Subbagging [AEEP02] constructs each bag r by sampling $m \leq n$ indices $r = (i_1, \dots, i_m)$ uniformly without replacement from $[n]$.*

The following result [SBW24b] ensures tail stability for any bootstrapped or subbagged algorithm:

Theorem 8 ([SBW24b]). *For any base learning algorithm \mathcal{A} returning outputs in Δ_{L-1} , the bagged algorithm $\tilde{\mathcal{A}}_m$ has tail stability (ε, δ) for any $\varepsilon, \delta > 0$ satisfying (2).*

Computational cost of bagging. In this section, we have worked with the ideal, derandomized version of bagging for simplicity—that is, we assume that the expected value $\mathbb{E}_r[\mathcal{A}(\mathcal{D}^r)(x)]$ is calculated exactly with respect to the distribution of r . In practice, of course, this is computationally infeasible (since for bootstrapping, say, there are n^m possible bags r), and so we typically resort to Monte Carlo sampling to approximate this expected value, defining $\tilde{\mathcal{A}}_m(\mathcal{D})(x) = \frac{1}{B} \sum_{b=1}^B \mathcal{A}(\mathcal{D}^{r_b})(x)$, for B i.i.d. bags r_1, \dots, r_B sampled via bootstrapping or subbagging. See Appendix C for extensions of our main result, Theorem 6, to the finite- B version of the method.

3.2 The inflated argmax

We now return to the inflated argmax operator, as defined in Definition 1. This operator, in place of the standard argmax, allows for stability in classification.

While $\text{argmax}^\varepsilon$ is defined as an operator on \mathbb{R}^L , in the context of classification algorithms, we only apply it to vectors w lying in the simplex Δ_{L-1} (in particular, in the context of a two-stage classification procedure as in Section 1.1, we will apply the inflated argmax to the vector $w = \hat{p}(x)$ of estimated probabilities). In Figure 1, we visualize the inflated argmax applied to the simplex in a setting with three possible labels, $L = 3$. Each different shaded area corresponds to the region of the simplex Δ_{L-1} where $\text{argmax}^\varepsilon(w)$ returns a particular subset.

3.2.1 Inflated argmax leads to selection stability

Unlike the standard argmax, the inflated argmax allows us to achieve stability in the context of classification. This is due to the following theorem, which shows that the inflated argmax is ε -compatible, as introduced in Definition 4. (The proofs of this result and all properties of the inflated argmax are deferred to Appendix A.)

Theorem 9. *For any $\varepsilon > 0$ and any $v, w \in \mathbb{R}^L$, if $\|v - w\| < \varepsilon$ then $\text{argmax}^\varepsilon(v) \cap \text{argmax}^\varepsilon(w) \neq \emptyset$. In particular, viewing $\text{argmax}^\varepsilon$ as a map on the simplex Δ_{L-1} , $\text{argmax}^\varepsilon$ is ε -compatible.*

To gain a more concrete understanding of this theorem, we can look again at Figure 1 to examine the case $L = 3$. Theorem 9 ensures that any two regions corresponding to disjoint subsets—e.g., the top corner region corresponding to $\{2\}$, and the bottom center region corresponding to $\{1, 3\}$ —are distance at least ε apart. In particular, the region in the center, corresponding to vectors w that map to the full set $\{1, 2, 3\}$, is a curved triangle of constant width ε , also known as a *Reuleaux triangle* [Reu76; Wei05].

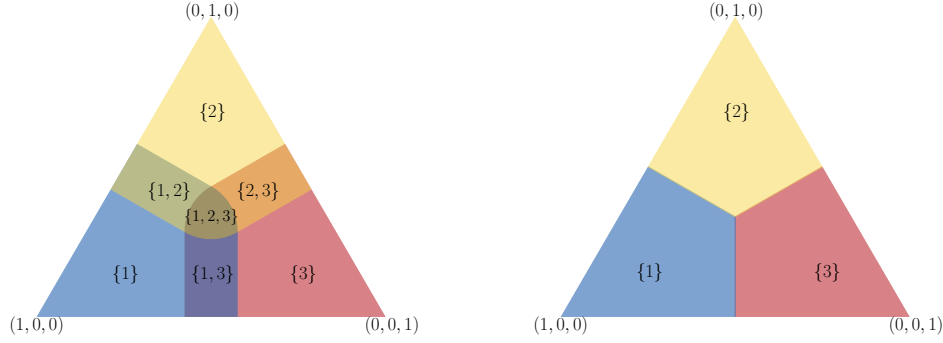


Figure 1: The left plot illustrates the inflated argmax (1) over the simplex Δ_{L-1} when $L = 3$. The numbers in brackets correspond to the output of the inflated argmax, $\text{argmax}^\varepsilon(w)$, for various points w in the simplex. The right plot shows the same but for the standard argmax, which corresponds to the limit of $\text{argmax}^\varepsilon(w)$ as $\varepsilon \rightarrow 0$.

Proof of Theorem 6. With the above results in place, we can now see that the inflated argmax allows us to achieve selection stability, when combined with a bagged algorithm. In particular, combining Theorem 8, Theorem 9, and Proposition 5 immediately implies our main result, Theorem 6.

3.2.2 Additional properties of the inflated argmax

The following result establishes some natural properties obeyed by the inflated argmax, and also compares to the standard argmax.

Proposition 10 (Basic properties of the inflated argmax). *Fix any $\varepsilon > 0$. The inflated argmax operator satisfies the following properties:*

1. (Including the argmax.) For any $w \in \mathbb{R}^L$, $\text{argmax}(w) \subseteq \text{argmax}^\varepsilon(w)$.³ Moreover, $\bigcap_{\varepsilon > 0} \text{argmax}^\varepsilon(w) = \text{argmax}(w)$.
2. (Monotonicity in ε .) For any $w \in \mathbb{R}^L$ and any $\varepsilon < \varepsilon'$, $\text{argmax}^\varepsilon(w) \subseteq \text{argmax}^{\varepsilon'}(w)$.
3. (Monotonicity in w .) For any $w \in \mathbb{R}^L$, if $w_j \leq w_k$, then $j \in \text{argmax}^\varepsilon(w) \Rightarrow k \in \text{argmax}^\varepsilon(w)$.
4. (Permutation invariance.) For any $v, w \in \mathbb{R}^L$, if $v = (w_{\sigma(1)}, \dots, w_{\sigma(L)})$ for some permutation σ on $[L]$, then $j \in \text{argmax}^\varepsilon(v) \Leftrightarrow \sigma(j) \in \text{argmax}^\varepsilon(w)$.

Next, while we have established that inflated argmax offers favorable stability properties, we have not yet asked whether it can be efficiently computed—in particular, it is not immediately clear how to verify the condition $\text{dist}(w, R_j^\varepsilon) < \varepsilon$ in (1), in order to determine whether $j \in \text{argmax}^\varepsilon(w)$. The following result offers an efficient algorithm for computing the inflated argmax set.

Proposition 11 (Computing the inflated argmax). *Fix any $w \in \mathbb{R}^L$ and $\varepsilon > 0$. Let $w_{[1]} \geq \dots \geq w_{[L]}$ denote the order statistics of w , and define*

$$\hat{k}(w) = \max \left\{ k \in [L] : \left(\sum_{\ell=1}^k (w_{[\ell]} - w_{[k]}) \right)^2 + \sum_{\ell=1}^k (w_{[\ell]} - w_{[k]})^2 \leq \varepsilon^2 \right\}.$$

Then

$$\text{argmax}^\varepsilon(w) = \left\{ j \in [L] : w_j > \frac{\varepsilon}{\sqrt{2}} + \hat{A}_1(w) - \sqrt{\hat{k}(w) + 1} \sqrt{\frac{\varepsilon^2}{\hat{k}(w)} + (\hat{A}_1(w))^2 - \hat{A}_2(w)} \right\},$$

where $\hat{A}_1(w) = \frac{w_{[1]} + \dots + w_{[\hat{k}(w)]}}{\hat{k}(w)}$, and $\hat{A}_2(w) = \frac{w_{[1]}^2 + \dots + w_{[\hat{k}(w)]}^2}{\hat{k}(w)}$.

³In this result, $\text{argmax}(w)$ should be interpreted as the set of *all* maximizing entries of w —i.e., in the case of ties, $\text{argmax}(w)$ may not be a singleton set.

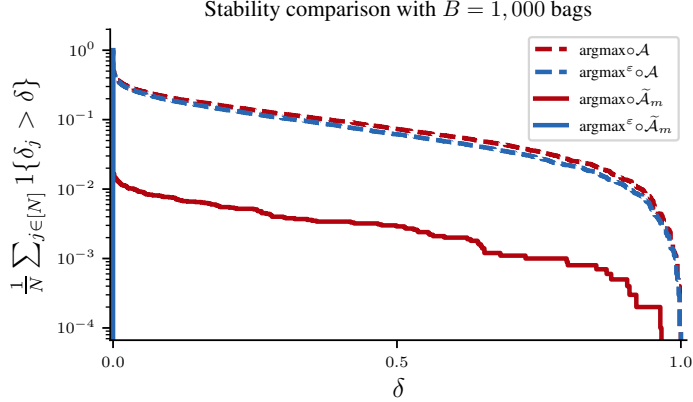


Figure 2: Results on the Fashion MNIST data set. The figure shows the instability δ_j (defined in (4)) over all test points $j = 1, \dots, N$. The curves display the fraction of δ_j 's that exceed δ , for each value $\delta \in [0, 1]$. The vertical axis is on a log scale. See Section 4 for details.

Finally, thus far we have established that $\text{argmax}^\varepsilon$ enables us to achieve stable classification with a computationally efficient selection rule—but we do not yet know whether $\text{argmax}^\varepsilon$ is optimal, or whether some other choice of s might lead to smaller output sets \hat{S} (while still offering assumption-free stability). For instance, we might consider a fixed-margin rule,

$$s_{\text{margin}}^\varepsilon(w) = \{j : w_j > \max_\ell w_\ell - \varepsilon/\sqrt{2}\}, \quad (3)$$

for which ε -compatibility also holds—might this simpler rule be better than the inflated argmax ? Our next result establishes that—under some natural constraints— $\text{argmax}^\varepsilon$ is in fact the optimal choice of selection rule, in the sense of returning a singleton set (i.e., $|\hat{S}| = 1$) as often as possible.

Proposition 12 (Optimality of the inflated argmax). *Let $s : \Delta_{L-1} \rightarrow \mathcal{P}([L])$ be any selection rule. Suppose s is ε -compatible (Definition 4), permutation invariant (in the sense of Proposition 10), and contains the argmax . Then for any $w \in \Delta_{L-1}$ and any $j \in [L]$,*

$$s(w) = \{j\} \implies \text{argmax}^\varepsilon(w) = \{j\}.$$

In other words, for any selection rule s satisfying the assumptions of the proposition (which includes the fixed-margin rule, $s_{\text{margin}}^\varepsilon$), if $s(w)$ is a singleton set then so is $\text{argmax}^\varepsilon(w)$. (See also Appendix D for a closer comparison between the inflated argmax and the fixed-margin selection rule given in (3).)

4 Experiments

In this section, we evaluate our proposed pipeline, combining subbagging with the inflated argmax , with deep learning models and on a common benchmark data set.⁴

Data and models. We use Fashion-MNIST [XRV17], which consists of $n = 60,000$ training pairs (X_i, Y_i) , $N = 10,000$ test pairs $(\tilde{X}_j, \tilde{Y}_j)$, and $L = 10$ classes. For each data point (X, Y) , X is a 28×28 grayscale image that pictures a clothing item, and $Y \in [L]$ indicates the type of item, e.g., a dress, a coat, etc. The base model we use is a variant of LeNet-5, implemented in PyTorch [PGML19] tutorials as `GarmentClassifier()`. The base algorithm \mathcal{A} trains this classifier using 5 epochs of stochastic gradient descent.

Methods and evaluation. We compare four methods:

1. The argmax of the base learning algorithm \mathcal{A} .

⁴Code to fully reproduce the experiment is available at <https://github.com/jake-soloff/stable-argmax-experiments>. Training all of the models for this experiment took a total of four hours on 10 CPUs running in parallel on a single computing cluster.

Selection rule	Algo.	$\beta_{\text{correct-single}} \nearrow$	$\beta_{\text{set-size}} \searrow$	$\beta_{\text{max-instability}} \searrow$
argmax	\mathcal{A}	0.879 (0.003)	1.000 (0.000)	1.000
argmax $^\varepsilon$	\mathcal{A}	0.873 (0.003)	1.015 (0.001)	1.000
argmax	$\tilde{\mathcal{A}}_m$	0.893 (0.003)	1.000 (0.000)	0.966
argmax $^\varepsilon$	$\tilde{\mathcal{A}}_m$	0.886 (0.003)	1.018 (0.001)	0.000

Table 1: Results on the Fashion MNIST data set. We display the frequency of returning the correct label as a singleton, $\beta_{\text{correct-single}}$ and the average size, $\beta_{\text{set-size}}$, both of which are defined in (6). Standard errors for these averages are shown in parentheses. The final column shows the worst-case instability over the test set, $\beta_{\text{max-instability}}$, defined in (5). For each metric, the symbol \nearrow indicates that higher values are desirable, while \searrow indicates that lower values are desirable.

2. The ε -inflated argmax of the base learning algorithm \mathcal{A} with tolerance $\varepsilon = .05$.
3. The argmax of the subbagged algorithm $\tilde{\mathcal{A}}_m$, with $B = 1,000$ bags of size $m = n/2$.
4. The ε -inflated argmax of the subbagged algorithm $\tilde{\mathcal{A}}_m$, with $B = 1,000$ bags of size $m = n/2$ and tolerance $\varepsilon = .05$.

We evaluate each method based on several metrics. First, to assess selection stability, for each test point $j = 1, \dots, N$ we compute its instability

$$\delta_j := \frac{1}{500} \sum_{k=1}^{500} \mathbf{1} \left\{ s(\hat{p}(\tilde{X}_j)) \cap s(\hat{p}^{\setminus i_k}(\tilde{X}_j)) = \emptyset \right\}, \quad (4)$$

where i_1, \dots, i_{500} are sampled uniformly without replacement from $[n]$ (i.e., we are sampling 500 leave-one-out models $\hat{p}^{\setminus i}$ to compare to the model \hat{p} trained on the full training set). Since our theory concerns worst-case instability over all possible test points, we evaluate the maximum instability

$$\beta_{\text{max-instability}} := \max_{j \in [N]} \delta_j. \quad (5)$$

Second, to evaluate how accurate each method is, we compute how often the method returns the correct label as a singleton $\beta_{\text{correct-single}}$ and set size $\beta_{\text{set-size}}$ (the number of labels in the candidate set):

$$\beta_{\text{correct-single}} := \frac{1}{N} \sum_{j=1}^N \mathbf{1} \left\{ s(\hat{p}(\tilde{X}_j)) = \{\tilde{Y}_j\} \right\}, \quad \beta_{\text{set-size}} := \frac{1}{N} \sum_{j=1}^N |s(\hat{p}(\tilde{X}_j))|. \quad (6)$$

Ideally we would want a method to return the correct singleton as frequently as possible (a large value of $\beta_{\text{correct-single}} \in [0, 1]$ that is close to 1) and small set size (a value of $\beta_{\text{set-size}} \geq 1$ that is close to 1).

Results. In Figure 2, we present results for the instability of each method, plotting the survival function of the instability for all test points $(\delta_j)_{j \in [N]}$. The standard argmax applied to the base algorithm, $\text{argmax} \circ \mathcal{A}$, has the longest tail, meaning δ_j is large for many test points j . The inflated argmax applied to the base algorithm, $\text{argmax}^\varepsilon \circ \mathcal{A}$, offers only a very small improvement on the stability. By contrast, applying the standard argmax to the subbagged algorithm, $\text{argmax} \circ \tilde{\mathcal{A}}_m$, offers a much more substantial improvement, since the red dashed curve is much smaller than both of the solid curves. Still, some of the largest δ_j for this method are near 1, meaning for these test points the returned set is sensitive to dropping a single training instance. Combining the inflated argmax with subbagging, $\text{argmax}^\varepsilon \circ \tilde{\mathcal{A}}_m$, offers a dramatic improvement: in this case $\delta_j = 0$ for every $j = 1, \dots, N$.

In Table 1, we present the average measures, $\beta_{\text{correct-single}}$ and $\beta_{\text{set-size}}$ and the worst-case measure $\beta_{\text{max-instability}}$. For both the base algorithm and the subbagged algorithm, applying the inflated argmax incurs a small cost both in terms of the frequency of returning the correct label as a singleton and the average size. The inflated argmax increases set size only very slightly, but when combined with subbagging, we improve upon the $\beta_{\text{correct-single}}$ of the original method and achieve an extremely high level of stability ($\beta_{\text{max-instability}} = 0$), while returning a singleton set in the vast majority of cases.

We extend our experiment in Appendix E. In particular, we compare the inflated argmax to some existing methods for set-valued classification, including simple thresholding and top- K rules as well as more involved methods, all of which are Bayes rules for various utility functions [MWDH21]. We also evaluate each method using some additional metrics from set-valued classification, including utility-discounted predictive accuracy [ZCM12].

5 Discussion

Related work. There is an enormous literature on set-valued classification, so we only reference some of the most relevant works. Much prior work has considered the possibility of a more relaxed argmax to allow for uncertainty [Bri90; BV16; MA16; BMN20; ZLYW23]. The more recent papers in this line of work have focused on producing a sparse set of weights, but none of these works offer a formal stability guarantee for the support of the weights. Our work is the first to propose a version of the argmax that can provably control a notion of stability for the classification setting.

Our definition of selection stability relies on a notion of consistency between sets—two sets of candidate labels are consistent if they have at least one common element. This is similar in flavor to conformal classification [Lei14; SLW19], where a set of candidate values is valid if it contains the correct label; this method does not rely on any distributional assumptions, and consequently has been applied to complex and high-dimensional applications such as generative language models [QFSY23]. These frameworks share the motivation of avoiding ‘overconfidence in the assignment of a definite label’ [HPW18].

Overview, limitations and open questions. We prove a guarantee on the selection stability based on our methodology combining bagging with the inflated argmax. Theorem 6 does not place any assumptions on the learning algorithm nor on the data, including any distributional assumptions. In fact, the training data and test point may be chosen adversarially based on the algorithm, and the output will still be stable. Moreover, we do not assume that the sample size is large: the guarantee holds for any fixed training set size n and improves as n increases. Furthermore, the inflated argmax selection rule ensures that the returned sets of candidate labels are as small as possible (i.e., that there is as little ambiguity as possible about the predicted label for any given test point x).

While our theorem does not require assumptions, our method does require bagging the base learning algorithm. The main limitation of our work is that bagging is computationally intensive. However, training different bags is easily parallelizable, which is what allowed us to easily train a convolutional neural network on $B = 1,000$ total subsets of the training data. Moreover, while the original definition of bagging uses the conventional bootstrap, where each bag contains as many samples as the original data set, i.e. $m = n$, in our framework we allow for arbitrary bag size m , which could be much smaller than the sample size n . Massively subsampling the data ($m \ll n$) can actually help scale learning algorithms to large data sets [KTSJ14]. Moreover, bagging with $m = n$ can be expensive, but there are still many areas of machine learning where it is used, notably in Random Forests. Finally, our experiments also show that a modest number of bags ($B = 1,000$) is all that we really need to start seeing major gains in selection stability.

We leave open several avenues for future work. Practitioners may be interested in other forms of discrete stability, which are more strict than requiring one common element between \hat{S} and \hat{S}^i . For instance, one popular way to measure set similarity is the Jaccard index, given by $\frac{|\hat{S} \cap \hat{S}^i|}{|\hat{S} \cup \hat{S}^i|}$. Another open problem is to extend the stability framework and methods studied here to more structured discrete outputs, such as in clustering and variable selection, where nontrivial metrics on the output are more common.

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation via grant DMS-2023109 DMS-2023109. R.F.B. and J.A.S. gratefully acknowledge the support of the Office of Naval Research via grant N00014-20-1-2337. R.M.W. gratefully acknowledges the support of the NSF-Simons National Institute for Theory and Mathematics in Biology (NSF DMS-2235451, Simons Foundation MP-TMPS-00005320), and AFOSR FA9550-18-1-0166. The authors thank Melissa Adrian for help with running the experiments.

References

- [AB23] Anastasios N Angelopoulos, Stephen Bates, et al. “Conformal prediction: A gentle introduction”. In: *Foundations and Trends® in Machine Learning* 16.4 (2023), pp. 494–591.
- [AEEP02] Savina Andonova et al. “A simple algorithm for learning stable machines”. In: *European Conference on Artificial Intelligence (ECAI)*. 2002, pp. 513–517.
- [BE02] Olivier Bousquet and André Elisseeff. “Stability and generalization”. In: *The Journal of Machine Learning Research* 2 (2002), pp. 499–526.
- [BMN20] Mathieu Blondel, André F. T. Martins, and Vlad Niculae. “Learning with Fenchel-Young losses”. In: *J. Mach. Learn. Res.* 21 (2020), Paper No. 35, 69. ISSN: 1532-4435,1533-7928.
- [Bre96a] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.
- [Bre96b] Leo Breiman. “Heuristics of instability and stabilization in model selection”. In: *The Annals of Statistics* 24.6 (1996), pp. 2350–2383.
- [Bri90] John S. Bridle. “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition”. In: *Neurocomputing (Les Arcs, 1989)*. Vol. 68. NATO Adv. Sci. Inst. Ser. F: Comput. Systems Sci. Springer, Berlin, 1990, pp. 227–236. ISBN: 3-540-53278-1.
- [BV16] Alexandre de Brébisson and Pascal Vincent. “An exploration of softmax alternatives belonging to the spherical loss family”. In: *International Conference on Learning Representations (ICLR)*. 2016.
- [CDHL21] Evgenii Chzhen et al. “Set-valued classification—overview via a unified framework”. In: *arXiv preprint arXiv:2102.12318* (2021).
- [CMX11] Constantine Caramanis, Shie Mannor, and Huan Xu. “Robust optimization in machine learning”. In: (2011).
- [DDB09] Juan José Del Coz, Jorge Díez, and Antonio Bahamonde. “Learning Nondeterministic Classifiers.” In: *J. Mach. Learn. Res.* 10.10 (2009).
- [FSVC19] Sorelle A Friedler et al. “A comparative study of fairness-enhancing interventions in machine learning”. In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 329–338.
- [Gry93] Eugen Grycko. “Classification with set-valued decision functions”. In: *Information and Classification: Concepts, Methods and Applications Proceedings of the 16th Annual Conference of the “Gesellschaft für Klassifikation eV” University of Dortmund, April 1–3, 1992*. Springer. 1993, pp. 218–224.
- [HPW18] Yotam Hechtlinger, Barnabás Póczos, and Larry Wasserman. “Cautious deep learning”. In: *arXiv preprint arXiv:1805.09460* (2018).
- [HV19] Lingxiao Huang and Nisheeth Vishnoi. “Stable and fair classification”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2879–2890.
- [KTSJ14] Ariel Kleiner et al. “A scalable bootstrap for massive data”. In: *Journal of the Royal Statistical Society: Series B: Statistical Methodology* (2014), pp. 795–816.
- [Lei14] Jing Lei. “Classification with confidence”. In: *Biometrika* 101.4 (2014), pp. 755–769. ISSN: 0006-3444,1464-3510. DOI: 10.1093/biomet/asu038. URL: <https://doi.org/10.1093/biomet/asu038>.
- [MA16] Andre Martins and Ramon Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”. In: *International conference on machine learning*. PMLR. 2016, pp. 1614–1623.
- [MSKA19] W James Murdoch et al. “Definitions, methods, and applications in interpretable machine learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (2019), pp. 22071–22080.
- [MWDH21] Thomas Mortier et al. “Efficient set-valued prediction in multi-class classification”. In: *Data Mining and Knowledge Discovery* 35.4 (2021), pp. 1435–1469.
- [NDMH18] VL Nguyen et al. “Reliable multi-class classification based on pairwise epistemic and aleatoric uncertainty”. In: *27th international joint conference on artificial intelligence (IJCAI)*. 2018, pp. 5089–5095.

- [PGML19] Adam Paszke et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).
- [PPVG02] Harris Papadopoulos et al. “Inductive confidence machines for regression”. In: *Machine learning: ECML 2002: 13th European conference on machine learning Helsinki, Finland, August 19–23, 2002 proceedings 13*. Springer, 2002, pp. 345–356.
- [QFSY23] Victor Quach et al. “Conformal language modeling”. In: *arXiv preprint arXiv:2306.10193* (2023).
- [Reu76] F Reuleaux. *Kinematics of machinery: outlines of a theory of machines*. Reprinted as “The Kinematics of Machinery”. New York: Dover, 1876.
- [SBW24a] Jake A. Soloff, Rina Foygel Barber, and Rebecca Willett. “Bagging Provides Assumption-free Stability”. In: *Journal of Machine Learning Research* 25.131 (2024), pp. 1–35. URL: <http://jmlr.org/papers/v25/23-0536.html>.
- [SBW24b] Jake A. Soloff, Rina Foygel Barber, and Rebecca Willett. “Stability via resampling: statistical problems beyond the real line”. In: *arXiv preprint arXiv:2405.09511* (2024).
- [SLW19] Mauricio Sadinle, Jing Lei, and Larry Wasserman. “Least ambiguous set-valued classifiers with bounded error levels”. In: *J. Amer. Statist. Assoc.* 114.525 (2019), pp. 223–234. ISSN: 0162-1459,1537-274X. DOI: 10.1080/01621459.2017.1395341. URL: <https://doi.org/10.1080/01621459.2017.1395341>.
- [VGS05] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, New York, 2005, pp. xvi+324.
- [Wei05] Eric W Weisstein. “Reuleaux Triangle”. In: *MathWorld—A Wolfram Web Resource* (2005). <https://mathworld.wolfram.com/ReuleauxTriangle.html>.
- [XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: 1708.07747 [cs.LG].
- [YB24] Bin Yu and Rebecca Barter. *Veridical Data Science: The Practice of Responsible Data Analysis and Decision Making*. MIT Press, 2024.
- [YK20] Bin Yu and Karl Kumbier. “Veridical data science”. In: *Proceedings of the National Academy of Sciences* 117.8 (2020), pp. 3920–3929.
- [ZCM12] Marco Zaffalon, Giorgio Corani, and Denis Mauá. “Evaluating credal classifiers by utility-discounted predictive accuracy”. In: *Internat. J. Approx. Reason.* 53.8 (2012), pp. 1282–1301. ISSN: 0888-613X,1873-4731. DOI: 10.1016/j.ijar.2012.06.022. URL: <https://doi.org/10.1016/j.ijar.2012.06.022>.
- [ZLYW23] Shuai Zhao et al. “From softmax to nucleusmax: A novel sparse language model for chinese radiology report summarization”. In: *ACM Transactions on Asian and Low-Resource Language Information Processing* 22.6 (2023), pp. 1–21.

A Proofs of properties of the inflated argmax

In this section, we prove all the results of Section 3.2, establishing the various properties of the inflated argmax. In fact, all the proofs will rely on the following theorem, which gives an alternative characterization of the set $\operatorname{argmax}^\varepsilon(w)$.

Theorem 13. Fix $\varepsilon > 0$. Define the map $c_\varepsilon : \mathbb{R}^L \rightarrow \mathbb{R}$ as

$$c_\varepsilon(w) \text{ is the unique solution to } \left(\sum_{j \in [L]} (w_j - c)_+ \right)^2 + \sum_{j \in [L]} (w_j - c)_+^2 = \varepsilon^2,$$

and define the map $t_\varepsilon : \mathbb{R}^L \rightarrow \mathbb{R}$ as

$$t_\varepsilon(w) = c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{j \in [L]} (w_j - c_\varepsilon(w))_+.$$

Then for any $w \in \mathbb{R}^L$, it holds that

$$\operatorname{argmax}^\varepsilon(w) = \{j \in [L] : w_j > t_\varepsilon(w)\}.$$

Moreover, $t_\varepsilon(w) \leq c_\varepsilon(w)$ for all w and all ε , and $t_\varepsilon(w)$ and $c_\varepsilon(w)$ are both nondecreasing functions of ε , and both nondecreasing functions of w (i.e., if $w \leq w'$ holds coordinatewise, then $c_\varepsilon(w) \leq c_\varepsilon(w')$ and $t_\varepsilon(w) \leq t_\varepsilon(w')$).

With this key result in place, we are now ready to turn to the proofs of the results stated in Section 3.2. Throughout all the proofs below, the functions $c_\varepsilon(w)$ and $t_\varepsilon(w)$ are defined as in the statement of Theorem 13.

A.1 Proof of Theorem 9

Let $w, w' \in \mathbb{R}^L$ be any vectors with $\operatorname{argmax}^\varepsilon(w) \cap \operatorname{argmax}^\varepsilon(w') = \emptyset$. Define

$$B = \{j \in [L] : w_j > c_\varepsilon(w)\}, \quad B' = \{j \in [L] : w'_j > c_\varepsilon(w')\},$$

and note that we must have $B \subseteq \operatorname{argmax}^\varepsilon(w)$ since for $j \notin \operatorname{argmax}^\varepsilon(w)$, $w_j \leq t_\varepsilon(w) \leq c_\varepsilon(w)$ by Theorem 13. Similarly, $B' \subseteq \operatorname{argmax}^\varepsilon(w')$. We also must have B (and similarly B') nonempty by definition of $c_\varepsilon(w)$.

For convenience, define $c = c_\varepsilon(w)$, $c' = c_\varepsilon(w')$, $t = t_\varepsilon(w)$, $t' = t_\varepsilon(w')$. We then have

$$\begin{aligned} \|w - w'\|^2 &\geq \|w_B - w'_B\|^2 + \|w'_{B'} - w_{B'}\|^2 \\ &= \|(w_B - c\mathbf{1}_B) + (t'\mathbf{1}_B - w'_B) + (c - t')\mathbf{1}_B\|^2 + \|(w'_{B'} - c'\mathbf{1}_{B'}) + (t\mathbf{1}_{B'} - w_{B'}) + (c' - t)\mathbf{1}_{B'}\|^2, \end{aligned} \quad (7)$$

where the first step holds since $B \cap B' = \emptyset$ due to $\operatorname{argmax}^\varepsilon(w) \cap \operatorname{argmax}^\varepsilon(w') = \emptyset$. Next we will need a technical lemma.

Lemma 14. Let $k, k' \geq 1$ be integers and let $\varepsilon > 0$. Then for any $x, y \in \mathbb{R}_{\geq 0}^k$, $x', y' \in \mathbb{R}_{\geq 0}^{k'}$, and $r, r' \in \mathbb{R}$, if

$$(x^\top \mathbf{1}_k)^2 + \|x\|^2 = (x'^\top \mathbf{1}_{k'})^2 + \|x'\|^2 = \varepsilon^2 \quad (8)$$

and

$$x^\top \mathbf{1}_k + x'^\top \mathbf{1}_{k'} = r + r' + \varepsilon\sqrt{2}, \quad (9)$$

then

$$\|x + y + r\mathbf{1}_k\|^2 + \|x' + y' + r'\mathbf{1}_{k'}\|^2 \geq \varepsilon^2.$$

To apply this lemma, first we let $k = |B| \geq 1$ and $k' = |B'| \geq 1$, and define

$$x = w_B - c\mathbf{1}_B, \quad y = t'\mathbf{1}_B - w'_B, \quad x' = w'_{B'} - c'\mathbf{1}_{B'}, \quad y' = t\mathbf{1}_{B'} - w_{B'},$$

and note that all these vectors have nonnegative entries (specifically, x and x' are nonnegative by definition of B and B' , while y and y' are nonnegative by Theorem 13, using the fact that

$B \cap s(w') = B' \cap s(w) = \emptyset$). Then the condition (8) is satisfied by definition of $c = c_\varepsilon(w)$ and $c' = c_\varepsilon(w')$. Define also

$$r = c - t', \quad r' = c' - t.$$

and note that

$$\begin{aligned} r + r' &= (c - t) + (c' - t') = \sum_{j \in [L]} (w_j - c)_+ - \varepsilon/\sqrt{2} + \sum_{j \in [L]} (w'_j - c')_+ - \varepsilon/\sqrt{2} \\ &= x^\top \mathbf{1}_k + x'^\top \mathbf{1}_{k'} - \varepsilon\sqrt{2}, \end{aligned}$$

where the second step holds by definition of $t = t_\varepsilon(w)$ and $t' = t_\varepsilon(w')$. Therefore, (9) is also satisfied. Returning to (7), Lemma 14 then implies that $\|w - w'\|^2 \geq \varepsilon^2$, which completes the proof of the theorem.

A.2 Proof of Proposition 10

First we verify that $\operatorname{argmax}(w) \subseteq \operatorname{argmax}^\varepsilon(w)$. Let $j \in \operatorname{argmax}(w)$, i.e., $w_j = \max_\ell w_\ell$. Let $v = w + \mathbf{e}_j \cdot \varepsilon/\sqrt{2}$, where $\mathbf{e}_j = (0, \dots, 0, 1, 0, \dots, 0)$ is the j th canonical basis vector. Then $v \in R_j^\varepsilon$, and $\|w - v\| = \varepsilon/\sqrt{2} < \varepsilon$, so we have $\operatorname{dist}(w, R_j^\varepsilon) < \varepsilon$ and therefore $j \in \operatorname{argmax}^\varepsilon(w)$. We also need to verify that $\bigcap_{\varepsilon > 0} \operatorname{argmax}^\varepsilon(w) = \operatorname{argmax}(w)$, i.e., for $j \notin \operatorname{argmax}(w)$, there is some $\varepsilon > 0$ with $j \notin \operatorname{argmax}^\varepsilon(w)$. Let $k \in [L]$ be an index with $w_k > w_j$, and let $\varepsilon = \sqrt{2}(w_k - w_j)$. Then for any $v \in R_j^\varepsilon$, we have

$$\begin{aligned} \|w - v\|^2 &\geq (w_j - v_j)^2 + (w_k - v_k)^2 = (w_k - \varepsilon/\sqrt{2} - v_j)^2 + (w_k - v_k)^2 \\ &\geq \inf_{t \in \mathbb{R}} \left\{ \left(t - (\varepsilon/\sqrt{2} + (v_j - v_k)) \right)^2 + t^2 \right\} = \frac{1}{2}(\varepsilon/\sqrt{2} + (v_j - v_k))^2 \geq \varepsilon^2, \end{aligned}$$

where the last step holds since $v_j - v_k \geq \varepsilon/\sqrt{2}$. This means that $\operatorname{dist}(w, R_j^\varepsilon) \geq \varepsilon$ and so $j \notin \operatorname{argmax}^\varepsilon(w)$ at this value of $\varepsilon > 0$.

Next we check monotonicity in ε . Fix $\varepsilon < \varepsilon'$. Then for any $j \in [L]$,

$$j \in \operatorname{argmax}^\varepsilon(\varepsilon) \iff w_j > t_\varepsilon(w) \implies w_j > t_{\varepsilon'}(w) \iff j \in \operatorname{argmax}^{\varepsilon'}(w),$$

where each step holds by Theorem 13.

Next we verify monotonicity in w . If $w_j \leq w_k$, then by Theorem 13

$$j \in \operatorname{argmax}^\varepsilon(w) \iff w_j > t_\varepsilon(w) \implies w_k > t_\varepsilon(w) \iff k \in \operatorname{argmax}^\varepsilon(w).$$

Finally we check permutation invariance. Suppose $v = (w_{\sigma(1)}, \dots, w_{\sigma(L)})$ is a permutation of w . Then by construction, we have $c_\varepsilon(v) = c_\varepsilon(w)$ and $t_\varepsilon(v) = t_\varepsilon(w)$. In particular,

$$j \in \operatorname{argmax}^\varepsilon(v) \iff v_j > t_\varepsilon(v) \iff w_{\sigma(j)} > t_\varepsilon(w) \iff \sigma(j) \in \operatorname{argmax}^\varepsilon(w).$$

A.3 Proof of Proposition 11

Define a function f_w as

$$f_w(c) = \left(\sum_{j \in [L]} (w_j - c)_+ \right)^2 + \sum_{j \in [L]} (w_j - c)_+^2. \quad (10)$$

Then $\hat{k}(w)$ can equivalently be defined as

$$\hat{k}(w) = \max\{k \in [L] : f_w(w_{[k]}) \leq \varepsilon^2\}.$$

(Note that this maximum is well defined, since $f_w(w_{[1]}) = 0$ and so the set is nonempty.) Since $c \mapsto f_w(c)$ is strictly decreasing over $c \leq w_{[1]}$, we see that the solution $c_\varepsilon(w)$ to the equation

$f_w(c) = \varepsilon^2$ must satisfy $c_\varepsilon(w) \leq w_{[\hat{k}(w)]}$, and (if $\hat{k}(w) < L$) also $c_\varepsilon(w) > w_{[\hat{k}(w)+1]}$. In particular, this implies

$$\begin{aligned} \varepsilon^2 &= \left(\sum_{j \in [L]} (w_j - c_\varepsilon(w))_+ \right)^2 + \sum_{j \in [L]} (w_j - c_\varepsilon(w))_+^2 \\ &= \left(\sum_{j=1}^{\hat{k}(w)} (w_j - c_\varepsilon(w)) \right)^2 + \sum_{j=1}^{\hat{k}(w)} (w_j - c_\varepsilon(w))^2 \\ &= \left(\hat{k}(w) \cdot (\hat{A}_1(w) - c_\varepsilon(w)) \right)^2 + \left(\hat{k}(w) \hat{A}_2(w) - 2\hat{k}(w) \hat{A}_1(w) c_\varepsilon(w) + \hat{k}(w) c_\varepsilon(w)^2 \right) \\ &= \hat{k}(w) (\hat{k}(w) + 1) \left(c_\varepsilon(w)^2 - 2c_\varepsilon(w) \hat{A}_1(w) + \frac{\hat{k}(w) (\hat{A}_1(w))^2 + \hat{A}_2(w)}{\hat{k}(w) + 1} \right). \end{aligned}$$

This is a quadratic function of $c_\varepsilon(w)$, and is solved by

$$c_\varepsilon(w) = \hat{A}_1 - \sqrt{\frac{(\hat{A}_1(w))^2 - \hat{A}_2(w) + \frac{\varepsilon^2}{\hat{k}(w)}}{\hat{k}(w) + 1}}.$$

We also have

$$\begin{aligned} t_\varepsilon(w) &= c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{j \in [L]} (w_j - c_\varepsilon(w))_+ \\ &= c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{j=1}^{\hat{k}(w)} (w_{[j]} - c_\varepsilon(w)) \\ &= (\hat{k}(w) + 1) c_\varepsilon(w) + \frac{\varepsilon}{\sqrt{2}} - \hat{k}(w) \hat{A}_1(w). \end{aligned}$$

Plugging in our above expression for $c_\varepsilon(w)$, then,

$$t_\varepsilon(w) = \frac{\varepsilon}{\sqrt{2}} + \hat{A}_1(w) - \sqrt{\hat{k}(w) + 1} \sqrt{(\hat{A}_1(w))^2 - \hat{A}_2(w) + \frac{\varepsilon^2}{\hat{k}(w)}}.$$

Since $\operatorname{argmax}^\varepsilon(w) = \{j : w_j > t_\varepsilon(w)\}$ by Theorem 13, this completes the proof.

A.4 Proof of Proposition 12

First we will need a lemma:

Lemma 15. *Fix $w \in \mathbb{R}^L$. Then $w \in R_j^\varepsilon$ if and only if $\operatorname{argmax}^\varepsilon(w) = \{j\}$.*

For intuition on this result, we can look back at Figure 1—for instance, the blue region marked by $\{1\}$ illustrates the set of vectors $w \in \Delta_{L-1}$ such that $\operatorname{argmax}^\varepsilon(w) = \{1\}$, and according to this lemma, this is equal to the region R_1^ε .

Using this lemma, we now need to show that, for any ε -compatible and permutation invariant selection rule s , if $s(w) = \{j\}$ for some $w \in \Delta_{L-1}$ and $j \in [L]$, then $w \in R_j^\varepsilon$ for some j . First, we must have $j = \operatorname{argmax}(w)$, since s is assumed to contain the argmax . Fix any $k \neq j$, and define a vector $v \in \Delta_{L-1}$ by permuting w_j and w_k :

$$v_\ell = \begin{cases} w_k, & \ell = j, \\ w_j, & \ell = k, \\ w_\ell, & \ell \neq j, k. \end{cases}$$

Then by permutation invariance we have $s(v) = \{k\}$. Since s is ε -compatible, we therefore have

$$\varepsilon \leq \|w - v\| = \sqrt{(w_j - v_j)^2 + (w_k - v_k)^2 + \sum_{\ell \neq j, k} (w_\ell - v_\ell)^2} = \sqrt{2(w_j - w_k)^2}.$$

Therefore we have $w_j - w_k \geq \varepsilon/\sqrt{2}$ for all $k \neq j$, which proves $w \in R_j^\varepsilon$ and therefore $\operatorname{argmax}^\varepsilon(w) = \{j\}$.

A.5 Proof of Theorem 13

Step 1: verifying that $c_\varepsilon(w)$ is well defined. First we check that the solution $c_\varepsilon(w)$ exists and is unique. Fix w and define the function f_w as in (10). Note that $f_w(c) \equiv 0$ for $c \geq \max_i w_i$, and $c \mapsto f_w(c)$ is strictly decreasing over $c \leq \max_i w_i$, with $\lim_{c \rightarrow -\infty} f_w(c) = \infty$; therefore, a unique solution $c_\varepsilon(w)$ to the equation $f_w(c) = \varepsilon^2$ must exist.

Step 2: verifying $t_\varepsilon(w) \leq c_\varepsilon(w)$. Next we check that the claim $t_\varepsilon(w) \leq c_\varepsilon(w)$ must hold. First, by definition of $c_\varepsilon(w)$, we have

$$\varepsilon^2 = \left(\sum_{j \in [L]} (w_j - c_\varepsilon(w))_+ \right)^2 + \sum_{j \in [L]} (w_j - c_\varepsilon(w))_+^2 \leq 2 \left(\sum_{j \in [L]} (w_j - c_\varepsilon(w))_+ \right)^2,$$

where the inequality holds by the properties of the ℓ_1 and ℓ_2 norms. Therefore,

$$\sum_{j \in [L]} (w_j - c)_+ \geq \varepsilon / \sqrt{2},$$

which verifies that

$$t_\varepsilon(w) = c_\varepsilon(w) + \varepsilon / \sqrt{2} - \sum_{j \in [L]} (w_j - c_\varepsilon(w))_+ \leq c_\varepsilon(w).$$

Step 3: checking monotonicity. Now we turn to verifying the monotonicity properties of $t_\varepsilon(w)$ and $c_\varepsilon(w)$. First we check that these functions are nonincreasing in ε . First, since $f_w(c)$ is nonincreasing in c , and $c_\varepsilon(w)$ is the solution to $f_w(c) = \varepsilon^2$, this immediately implies that $c_\varepsilon(w)$ is nonincreasing in ε . Next we turn to $t_\varepsilon(w)$. Define

$$t'(c) = c + \frac{1}{\sqrt{2}} \sqrt{\left(\sum_{j \in [L]} (w_j - c)_+ \right)^2 + \sum_{j \in [L]} (w_j - c)_+^2 - \sum_{j \in [L]} (w_j - c)_+},$$

so that we have $t_\varepsilon(w) = t'(c_\varepsilon(w))$ by construction. We can verify that $c \mapsto t'(c)$ is nondecreasing, and therefore, $t_\varepsilon(w) = t'(c_\varepsilon(w)) \leq t'(c_{\varepsilon'}(w)) = t_{\varepsilon'}(w)$, where the inequality holds since $c_\varepsilon(w) \leq c_{\varepsilon'}(w)$.

Next we check that $t_\varepsilon(w)$ and $c_\varepsilon(w)$ are nondecreasing functions of w . Fix any $w \leq w'$ (where the inequality is coordinatewise, i.e., $w_j \leq w'_j$ for all $j \in [L]$). Since $w \mapsto f_w(c)$ is a nondecreasing function, we have $f_w(c) \leq f_{w'}(c)$ for all c . We therefore have $f_{w'}(c_\varepsilon(w)) \geq f_w(c_\varepsilon(w)) = \varepsilon^2 = f_{w'}(c_\varepsilon(w'))$. Since $c \mapsto f_{w'}(c)$ is nonincreasing, therefore, $c_\varepsilon(w') \geq c_\varepsilon(w)$. Next we consider t_ε . Let $t''_\varepsilon(c) = c + \varepsilon / \sqrt{2} - \sum_{j \in [L]} (w_j - c)_+$, which is a nondecreasing function of c . Then we have $t_\varepsilon(w) = t''_\varepsilon(c_\varepsilon(w)) \leq t''_\varepsilon(c_\varepsilon(w')) = t_\varepsilon(w')$.

Step 4: returning to the inflated argmax. Finally, fixing any $w \in \mathbb{R}^L$, we will prove that $\operatorname{argmax}^\varepsilon(w) = \{j : w_j > t_\varepsilon(w)\}$. First choose any $j \in [L]$ with $w_j > t_\varepsilon(w)$. We need to verify that $j \in \operatorname{argmax}^\varepsilon(w)$. Define $v \in \mathbb{R}^L$ with entries

$$v_k = \begin{cases} \max\{c_\varepsilon(w) + \varepsilon / \sqrt{2}, w_j\}, & k = j, \\ \min\{c_\varepsilon(w), w_k\}, & k \neq j. \end{cases}$$

By construction, we have $v \in R_j^\varepsilon$. We calculate

$$\begin{aligned}
\text{dist}(w, R_j^\varepsilon)^2 &\leq \|w - v\|^2 \\
&= \left(w_j - \max\{c_\varepsilon(w) + \varepsilon/\sqrt{2}, w_j\} \right)^2 + \sum_{k \neq j} (w_k - \min\{c_\varepsilon(w), w_k\})^2 \\
&= \left(c_\varepsilon(w) + \varepsilon/\sqrt{2} - w_j \right)_+^2 + \sum_{k \neq j} (w_k - c_\varepsilon(w))_+^2 \\
&< \left(c_\varepsilon(w) + \varepsilon/\sqrt{2} - t_\varepsilon(w) \right)^2 + \sum_{k \neq j} (w_k - c_\varepsilon(w))_+^2 \\
&\leq \left(\sum_{k \in [L]} (w_k - c_\varepsilon(w))_+ \right)^2 + \sum_{k \in [L]} (w_k - c_\varepsilon(w))_+^2 = \varepsilon^2,
\end{aligned}$$

where the last two steps hold by definition of $t_\varepsilon(w)$ and $c_\varepsilon(w)$, while the strict inequality holds since $t_\varepsilon(w) \leq c_\varepsilon(w) < c_\varepsilon(w) + \varepsilon/\sqrt{2}$ as established above, while $w_j > t_\varepsilon(w)$ by assumption. Therefore $j \in \text{argmax}^\varepsilon(w)$.

Now we check the converse. For this last step, we will need a lemma. The following result characterizes the projection of w to the set R_j^ε .

Lemma 16. *Fix any $w \in \mathbb{R}^L$ and any $j \in [L]$. Then there is a unique $a \in \mathbb{R}$ satisfying*

$$w_j = a + \varepsilon/\sqrt{2} - \sum_{k \neq j} (w_k - a)_+. \quad (11)$$

Moreover, defining the vector $v \in \mathbb{R}^L$ as

$$v_k = \begin{cases} a + \varepsilon/\sqrt{2}, & k = j, \\ a \wedge w_k, & k \neq j, \end{cases} \quad (12)$$

it holds that $v = \text{argmin}_{u \in R_j^\varepsilon} \|w - u\|$, i.e., v is the projection of w to the set R_j^ε .

Next suppose $w_j \leq t_\varepsilon(w)$. We need to verify that $j \notin \text{argmax}^\varepsilon(w)$. Let a and v be defined as in Lemma 16 above. We can compare the equation (11) to the definition of $t_\varepsilon(w)$,

$$t_\varepsilon(w) = c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{k \in [L]} (w_k - c_\varepsilon(w))_+ = c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{k \neq j} (w_k - c_\varepsilon(w))_+,$$

where the last step holds since $w_j \leq t_\varepsilon(w)$ by assumption, and $t_\varepsilon(w) \leq c_\varepsilon(w)$ as proved above. Since $c \mapsto c + \varepsilon/\sqrt{2} - \sum_{k \neq j} (w_k - c)_+$ is an increasing function, and $w_j \leq t_\varepsilon(w)$, this implies

$a \leq c_\varepsilon(w)$. We then calculate

$$\begin{aligned}
\text{dist}(w, R_j^\varepsilon)^2 &= \|w - v\|^2 \\
&= \left(w_j - \left(a + \varepsilon/\sqrt{2} \right) \right)^2 + \sum_{k \neq j} (w_k - a \wedge w_k)^2 \text{ by (12)} \\
&= \left(w_j - \left(a + \varepsilon/\sqrt{2} \right) \right)^2 + \sum_{k \neq j} (w_k - a)_+^2 \\
&= \left(\sum_{k \neq j} (w_k - a)_+ \right)^2 + \sum_{k \neq j} (w_k - a)_+^2 \text{ by (11)} \\
&\geq \left(\sum_{k \neq j} (w_k - c_\varepsilon(w))_+ \right)^2 + \sum_{k \neq j} (w_k - c_\varepsilon(w))_+^2 \text{ since } a \leq c_\varepsilon(w) \\
&= \left(\sum_{k \in [L]} (w_k - c_\varepsilon(w))_+ \right)^2 + \sum_{k \in [L]} (w_k - c_\varepsilon(w))_+^2 \text{ since } w_j \leq t_\varepsilon(w) \leq c_\varepsilon(w) \\
&= \varepsilon^2 \text{ by definition of } c_\varepsilon(w).
\end{aligned}$$

Therefore, $\text{dist}(w, R_j^\varepsilon) \geq \varepsilon$, and so $j \notin \text{argmax}^\varepsilon(w)$.

A.6 Proofs of technical lemmas

A.6.1 Proof of Lemma 14

First, since y is constrained to have nonnegative entries,

$$\|x + y + r\mathbf{1}_k\|^2 \geq \|(x + r\mathbf{1}_k)_+\|^2,$$

where for a vector $v = (v_1, \dots, v_L) \in \mathbb{R}^L$, we write $(v)_+$ to denote the vector with j th entry given by $(v_j)_+ = \max\{v_j, 0\}$ for each j . The analogous bound holds for $\|x' + y' + r'\mathbf{1}_{k'}\|^2$.

$$\|x + y + r\mathbf{1}_k\|^2 + \|x' + y' + r'\mathbf{1}_{k'}\|^2 \geq \|(x + r\mathbf{1}_k)_+\|^2 + \|(x' + r'\mathbf{1}_{k'})_+\|^2.$$

We now need to show that

$$\|(x + r\mathbf{1}_k)_+\|^2 + \|(x' + r'\mathbf{1}_{k'})_+\|^2 \geq \varepsilon^2.$$

Next let

$$\bar{r} = \frac{r + r'}{2} = \frac{x^\top \mathbf{1}_k + x'^\top \mathbf{1}_{k'} - \varepsilon\sqrt{2}}{2}, \quad \Delta = \frac{-r + r'}{2},$$

so that we can write

$$r = \bar{r} - \Delta, \quad r' = \bar{r} + \Delta$$

for some $\Delta \in \mathbb{R}$. We we therefore need to show $\inf_{\Delta \in \mathbb{R}} f(\Delta) \geq \varepsilon^2$, where

$$f(\Delta) = \|(x + (\bar{r} - \Delta)\mathbf{1}_k)_+\|^2 + \|(x' + (\bar{r} + \Delta)\mathbf{1}_{k'})_+\|^2.$$

First, we observe that we can restrict the range of Δ . Specifically, for any $\Delta \geq \max_{j \in [L]} x_j + \bar{r}$, we have $f(\Delta) = \|(x' + (\bar{r} + \Delta)\mathbf{1}_{k'})_+\|^2$, which is a nondecreasing function; therefore,

$$\inf_{\Delta \geq \max_{j \in [L]} x_j + \bar{r}} f(\Delta) = f\left(\max_{j \in [L]} x_j + \bar{r}\right),$$

meaning that we do not need to consider values of Δ beyond this upper bound. Applying a similar argument for a lower bound, we see that from this point on we only need to verify that

$$f(\Delta) \geq \varepsilon^2 \text{ for } -\max_{j \in [k']} x'_j - \bar{r} \leq \Delta \leq \max_{j \in [L]} x_j + \bar{r}.$$

Moreover, for any Δ , we have

$$(x_j + (\bar{r} - \Delta))^2 = (x_j + (\bar{r} - \Delta))_+^2 + (x_j + (\bar{r} - \Delta))_-^2 \leq (x_j + (\bar{r} - \Delta))_+^2 + (\bar{r} - \Delta)^2$$

for all j , by nonnegativity of x . Furthermore we must have

$$(x_j + (\bar{r} - \Delta))^2 = (x_j + (\bar{r} - \Delta))_+^2$$

for at least one $j \in [L]$ when $\Delta \leq \max_{j \in [L]} x_j + \bar{r}$ as specified above (i.e., because for j maximizing the entry x_j , the value $(x + (\bar{r} - \Delta)\mathbf{1}_k)_j$ is nonnegative). Therefore,

$$\|(x + (\bar{r} - \Delta)\mathbf{1}_k)_+\|^2 \geq \|x + (\bar{r} - \Delta)\mathbf{1}_k\|^2 - (k-1)(\bar{r} - \Delta)^2.$$

Similarly, we can show that

$$\|(x' + (\bar{r} + \Delta)\mathbf{1}_{k'})_+\|^2 \geq \|x' + (\bar{r} + \Delta)\mathbf{1}_{k'}\|^2 - (k' - 1)(\bar{r} + \Delta)^2.$$

We then calculate

$$\begin{aligned} f(\Delta) &= \|(x + (\bar{r} - \Delta)\mathbf{1}_k)_+\|^2 + \|(x' + (\bar{r} + \Delta)\mathbf{1}_{k'})_+\|^2 \\ &\geq \|x + (\bar{r} - \Delta)\mathbf{1}_k\|^2 - (k-1)(\bar{r} - \Delta)^2 + \|x' + (\bar{r} + \Delta)\mathbf{1}_{k'}\|^2 - (k' - 1)(\bar{r} + \Delta)^2 \\ &= \|x\|^2 + 2(\bar{r} - \Delta)x^\top \mathbf{1}_k + (\bar{r} - \Delta)^2 + \|x'\|^2 + 2(\bar{r} + \Delta)x'^\top \mathbf{1}_{k'} + (\bar{r} + \Delta)^2 \\ &= 2\varepsilon^2 - (x^\top \mathbf{1}_k)^2 - (x'^\top \mathbf{1}_{k'})^2 + 2(\bar{r} - \Delta)x^\top \mathbf{1}_k + 2(\bar{r} + \Delta)x'^\top \mathbf{1}_{k'} + 2\bar{r}^2 + 2\Delta^2, \end{aligned}$$

where the last step holds by (8).

Writing $z = x^\top \mathbf{1}_k$ and $z' = x'^\top \mathbf{1}_{k'}$ for convenience, we have

$$\begin{aligned} f(\Delta) &\geq 2\varepsilon^2 - z^2 - z'^2 + 2(\bar{r} - \Delta)z + 2(\bar{r} + \Delta)z' + 2\bar{r}^2 + 2\Delta^2 \\ &= 2\varepsilon^2 - z^2 - z'^2 + 2\left(\frac{z + z' - \varepsilon\sqrt{2}}{2} - \Delta\right)z + 2\left(\frac{z + z' - \varepsilon\sqrt{2}}{2} + \Delta\right)z' \\ &\quad + 2\left(\frac{z + z' - \varepsilon\sqrt{2}}{2}\right)^2 + 2\Delta^2 \\ &= \varepsilon^2 + 4\left(z - \varepsilon/\sqrt{2}\right)\left(z' - \varepsilon/\sqrt{2}\right) + 2\left(\Delta - \frac{z - z'}{2}\right)^2 \\ &\geq \varepsilon^2 + 4\left(z - \varepsilon/\sqrt{2}\right)\left(z' - \varepsilon/\sqrt{2}\right). \end{aligned}$$

where the second step holds because $\bar{r} = \frac{r+r'}{2} = \frac{z+z'-\varepsilon\sqrt{2}}{2}$ by (9).

Finally, we also have $x^\top \mathbf{1}_k = \|x\|_1 \geq \|x\|$, where the first step holds since x is nonnegative, and so by assumption (8) we have $z = x^\top \mathbf{1}_k \geq \varepsilon/\sqrt{2}$. Similarly, $z' \geq \varepsilon/\sqrt{2}$. Therefore, $4(z - \varepsilon/\sqrt{2})(z' - \varepsilon/\sqrt{2}) \geq 0$, and so we have shown that $f(\Delta) \geq \varepsilon^2$, as desired.

A.6.2 Proof of Lemma 15

First, suppose $w \in R_j^\varepsilon$. Then $\operatorname{argmax}(w) = \{j\}$, and so $j \in \operatorname{argmax}^\varepsilon(w)$ by Proposition 10. Next, let $c = w_j - \varepsilon/\sqrt{2}$. Then $w_\ell \leq c$ for all $\ell \neq j$, and so defining f_w as in (10), we have

$$f_w(c) = (w_j - c)^2 + (w_j - c)^2 = \varepsilon^2 \implies c_\varepsilon(w) = c = w_j - \varepsilon/\sqrt{2}.$$

Then

$$t_\varepsilon(w) = c_\varepsilon(w) + \varepsilon/\sqrt{2} - \sum_{\ell \in [L]} (w_\ell - c_\varepsilon(w))_+ = (w_j - \varepsilon/\sqrt{2}) + \varepsilon/\sqrt{2} - \varepsilon/\sqrt{2} = w_j - \varepsilon/\sqrt{2},$$

and so for all $\ell \neq j$, we have $w_\ell \leq w_j - \varepsilon/\sqrt{2} = t_\varepsilon(w)$ and thus $\ell \notin \operatorname{argmax}^\varepsilon(w)$. This verifies that $\operatorname{argmax}^\varepsilon(w) = \{j\}$.

To prove the converse, suppose $\operatorname{argmax}^\varepsilon(w) = \{j\}$. By Proposition 10, we then have $\operatorname{argmax}(w) = \{j\}$. Let $k \in \operatorname{argmax}_{\ell \neq j} w_\ell$, then to show $w \in R_j^\varepsilon$ it suffices to show that $w_j \geq w_k + \varepsilon/\sqrt{2}$. Define $v \in \mathbb{R}^L$ as

$$v_\ell = \begin{cases} w_k, & \ell = j, \\ w_k + \varepsilon/\sqrt{2}, & \ell = k, \\ w_\ell, & \ell \neq j, k. \end{cases}$$

Then $v \in R_k^\varepsilon$, and so we must have $\|w - v\| \geq \varepsilon$ since $k \notin \operatorname{argmax}^\varepsilon(w)$. We calculate

$$\|w - v\|^2 = (w_j - v_j)^2 + (w_k - v_k)^2 = (w_j - w_k)^2 + (\varepsilon/\sqrt{2})^2.$$

This implies that we must have $(w_j - w_k)^2 \geq \varepsilon^2/2$, which completes the proof.

A.6.3 Proof of Lemma 16

First we verify existence and uniqueness of a . Let

$$f(a) = w_j - a + \sum_{k \neq j} (w_k - a)_+.$$

Then $f : \mathbb{R} \rightarrow \mathbb{R}$ is a continuous and strictly decreasing bijection, so $f(a) = \varepsilon/\sqrt{2}$ must have a unique solution.

Next let a be this unique solution and let v be defined as in (12). Now we verify that v is the unique solution to the optimization problem

$$v = \operatorname{argmin}_{u \in \mathbb{R}^L} \left\{ \frac{1}{2} \|u - w\|^2 : u_j \geq u_k + \varepsilon/\sqrt{2} \text{ for all } k \neq j \right\},$$

which defines the projection of w to R_j^ε . By the Karush–Kuhn–Tucker (KKT) conditions for first-order optimality, it is sufficient to verify that

$$v - w - \sum_{k \neq j} \lambda_k (\mathbf{e}_j - \mathbf{e}_k) = 0, \quad (13)$$

where $\lambda_k \geq 0$ for all k , and $\lambda_k = 0$ for any inactive constraints, i.e., any k for which $v_j > v_k + \varepsilon/\sqrt{2}$. Let $\lambda_k = (w_k - a)_+$ for each $k \neq j$. Note that if $v_j > v_k + \varepsilon/\sqrt{2}$ (i.e., an inactive constraint), then we must have $w_k < a$ and so $\lambda_k = 0$, by construction of v . Then, for any $\ell \neq j$,

$$\left(v - w - \sum_{k \neq j} \lambda_k (\mathbf{e}_j - \mathbf{e}_k) \right)_\ell = v_\ell - w_\ell + \lambda_\ell = a \wedge w_\ell - w_\ell + (w_\ell - a)_+ = 0,$$

where \mathbf{e}_k denotes the k th canonical basis vector in \mathbb{R}^L . This proves that the ℓ th coordinate of the system of equations (13) holds for each $\ell \neq j$. Now we verify the j th coordinate. We have

$$\begin{aligned} \left(v - w - \sum_{k \neq j} \lambda_k (\mathbf{e}_j - \mathbf{e}_k) \right)_j &= v_j - w_j - \sum_{k \neq j} \lambda_k \\ &= a + \varepsilon/\sqrt{2} - w_j - \sum_{k \neq j} (w_k - a)_+ = \varepsilon/\sqrt{2} - f(a) = 0, \end{aligned}$$

by our choice of a as the solution to $f(a) = \varepsilon/\sqrt{2}$. This verifies the KKT conditions, and thus, v is the projection of w to R_j^ε as claimed.

B Extension to randomized algorithms

As mentioned in Section 2, in many applications it is common to use randomization in the construction of a classification procedure, in the learning algorithm \mathcal{A} and/or in the selection rule s . In this section we formalize this more general framework, and will see how our results apply.

First, in the non-random setting, a learning algorithm \mathcal{A} maps a data set $\mathcal{D} \in (\mathcal{X} \times [L])^n$ to a fitted probability estimate function $\hat{p} : \mathcal{X} \rightarrow \Delta_{L-1}$ —we write the regression procedure as $\hat{p} = \mathcal{A}(\mathcal{D})$. In the randomized setting, we also allow for external randomness, $\hat{p} = \mathcal{A}(\mathcal{D}; \xi)$, where $\xi \sim \text{Uniform}[0, 1]$ is a random seed (e.g., we might use ξ to randomly shuffle the training data when running stochastic gradient descent).

Next, in the non-random setting, a selection rule s is a map from Δ_{L-1} to subsets of $[L]$, resulting in a candidate set of labels $\hat{S} = s(\hat{p}(x))$. In the randomized setting (for instance, if s is the argmax but with ties broken at random), we instead include a random seed $\zeta \sim \text{Uniform}[0, 1]$, and write $\hat{S} = s(\hat{p}(x); \zeta)$. Formally, then, we have $s : \Delta_{L-1} \times [0, 1] \rightarrow \mathcal{P}([L])$. (Note that the selection rule proposed in this work—the inflated argmax—is itself *not* random.)

Combining the two stages of the procedure, then, the classification algorithm is given by $\mathcal{C} = s \circ \mathcal{A}$, where given a training set \mathcal{D} and a test point x , along with i.i.d. random seeds $\xi, \zeta \sim \text{Uniform}[0, 1]$, we return the candidate set of labels given by

$$\mathcal{C}(\mathcal{D}, x; \xi, \zeta) = s(\hat{p}(x); \zeta) \text{ where } \hat{p} = \mathcal{A}(\mathcal{D}; \xi).$$

Of course, we can derive the non-random setting as a special case, simply by designing \mathcal{A} and s to not depend on the random seeds ξ and ζ .

Now, how do the results of this paper extend to this randomized setting? First, we need to reconsider our definition of selection stability (Definition 2). We will now define it as

$$\frac{1}{n} \sum_{i=1}^n \Pr \left\{ \hat{S} \cap \hat{S}^{\setminus i} = \emptyset \right\} \leq \delta,$$

where $\hat{S} = \mathcal{C}(\mathcal{D}, x; \xi, \zeta)$ and $\hat{S}^{\setminus i} = \mathcal{C}(\mathcal{D}^{\setminus i}, x; \xi, \zeta)$, and where the probabilities are taken with respect to the distribution of the random seeds. We can also consider a notion of algorithmic stability (Definition 3) that allows for randomization: we say that \mathcal{A} has tail stability (ε, δ) if

$$\frac{1}{n} \sum_{i=1}^n \Pr \left\{ \|\hat{p}(x) - \hat{p}^{\setminus i}(x)\| \geq \varepsilon \right\} \leq \delta,$$

where $\hat{p} = \mathcal{A}(\mathcal{D}; \xi)$ and $\hat{p}^{\setminus i} = \mathcal{A}(\mathcal{D}^{\setminus i}; \xi)$, and where again probability is taken with respect to the distribution of ξ .

With these definitions in place, we observe that the result of Proposition 5 still holds in this more general setting for the selection rule $\text{argmax}^\varepsilon$: if \mathcal{A} is a randomized algorithm with tail stability (ε, δ) (under the new definition given above), then the randomized classification algorithm given by $\mathcal{C} = \text{argmax}^\varepsilon \circ \mathcal{A}$ has selection stability δ (again, under the new definition given above), simply due to the ε -compatibility property of $\text{argmax}^\varepsilon$ (Theorem 9). [SBW24b] proved Theorem 8 in this more general setting of randomized learning algorithms \mathcal{A} , so Theorem 6 holds for randomized \mathcal{A} under this more general definition of selection stability.

C Extension to finitely many bags

In Section 3.1, we discussed how in practice, the bagged algorithm $\tilde{\mathcal{A}}_m$ would be constructed by taking an empirical average over some large number B of sampled bags, rather than computing the exact expectation $\mathbb{E}_r[\mathcal{A}(\mathcal{D}^r)(x)]$. Now that we have allowed for randomized learning algorithms (as in Appendix B), we can formalize this setting: for a random seed $\xi \sim \text{Uniform}[0, 1]$, we define

$$\tilde{\mathcal{A}}_m(\mathcal{D}, \xi)(x) = \frac{1}{B} \sum_{b=1}^B \mathcal{A}(\mathcal{D}^{r_b})(x),$$

where the random draw of B many bags, r_1, \dots, r_B , is generated using the random seed ξ . Recall that [SBW24b] showed tail stability for the bagged version of any algorithm (as restated in Theorem 8)—their work also gives a result for the finite- B case, as follows:

Theorem 17 ([SBW24b]). *For any base learning algorithm \mathcal{A} returning outputs in Δ_{L-1} , the bagged algorithm $\tilde{\mathcal{A}}_m$ (computed with a finite number of bags, B) has tail stability (ε, δ) for any $\varepsilon, \delta > 0$ satisfying*

$$\delta = \varepsilon^{-2} \cdot (1 - 1/L) \left(\frac{1}{n-1} \cdot \frac{p_{n,m}}{1 - p_{n,m}} + \frac{16e^2}{B} \right).$$

In particular, combined with our result on the ε -compatibility of the inflated argmax , we have the following generalization of our main result (Theorem 6):

Theorem 18. *Fix any sample size n , any bag size m , any inflation parameter $\varepsilon > 0$, and any number of bags $B \geq 1$. For any base learning algorithm \mathcal{A} , the classification algorithm $\mathcal{C} = \text{argmax}^\varepsilon \circ \tilde{\mathcal{A}}_m$, obtained by combining the bagged version of \mathcal{A} (with B many bags) together with the inflated argmax , satisfies selection stability δ where*

$$\delta = \varepsilon^{-2} \cdot (1 - 1/L) \left(\frac{1}{n-1} \cdot \frac{p_{n,m}}{1 - p_{n,m}} + \frac{16e^2}{B} \right). \quad (14)$$

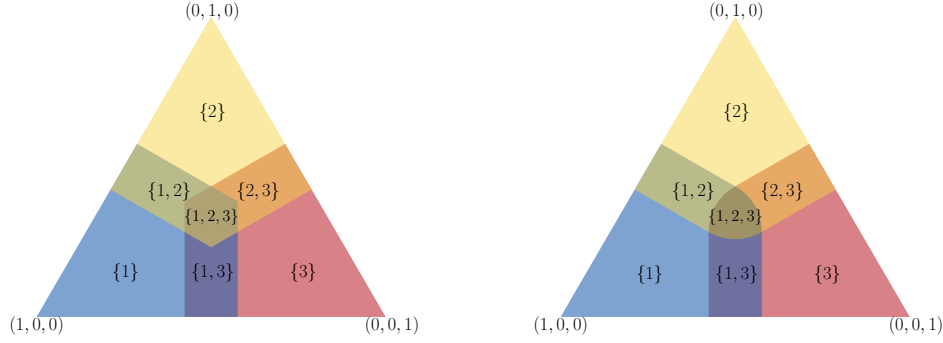


Figure 3: The fixed-margin selection rule (3), left, and the inflated argmax (1), right, when $L = 3$.

Comparing to our main result for the derandomized case (Theorem 6, which can be interpreted as taking $B \rightarrow \infty$), we see that this finite- B result is essentially the same as Theorem 6 once $B \gg n$.

D Compare to a simpler selection rule: the fixed-margin rule

In this section, we compare to the simpler fixed-margin selection rule, which was defined earlier in (3)—for convenience, we define it again here:

$$s_{\text{margin}}^\varepsilon(w) = \left\{ j \in [L] : w_j > \max_{\ell \in [L]} w_\ell - \varepsilon/\sqrt{2} \right\},$$

the set of all indices j that are within some margin of being the maximum.

In this section, we will see that this fixed-margin selection rule is ε -compatible. Since this rule clearly has the advantage of being much simpler than the inflated argmax (both in terms of its definition and interpretability, and in terms of its computation), we might ask whether this rule is perhaps preferable to the more complex inflated argmax. However, we will also see that the fixed-margin selection rule can be very inefficient compared to the inflated argmax: the inflated argmax can never return a larger set, and will often return a substantially smaller one.

D.1 Theoretical results

First, we verify that this simple rule satisfies ε -compatibility (Definition 4).

Proposition 19. *For any $\varepsilon > 0$, the selection rule $s_{\text{margin}}^\varepsilon$ is ε -compatible.*

In particular, since $s_{\text{margin}}^\varepsilon$ clearly contains the argmax (i.e., $\text{argmax}(w) \subseteq s_{\text{margin}}^\varepsilon(w)$) and satisfies permutation invariance (in the sense of Proposition 10), by Proposition 12 this immediately implies that

$$s_{\text{margin}}^\varepsilon(w) = \{j\} \implies \text{argmax}^\varepsilon(w) = \{j\}$$

for all $w \in \mathbb{R}^L$, i.e., $\text{argmax}^\varepsilon$ is at least as good as $s_{\text{margin}}^\varepsilon$ at returning a singleton set. However, for this particular selection rule, we can state an even stronger result:

Proposition 20. *For any $w \in \mathbb{R}^L$ and any $\varepsilon > 0$,*

$$s_{\text{margin}}^\varepsilon(w) \supseteq \text{argmax}^\varepsilon(w).$$

In particular, this theoretical result ensures that the set of candidate labels \hat{S} returned by the inflated argmax can never be larger than for the fixed-margin selection rule. In low dimensions, however, the improvement is small. Indeed, for $L = 2$, we actually have $s_{\text{margin}}^\varepsilon(w) = \text{argmax}^\varepsilon(w)$ for all w . For $L = 3$, Figure 3 shows that the inflated argmax is strictly better (i.e., the set inclusion result of Proposition 20 can, for certain values of w , be a strict inclusion), but the difference in this low-dimensional setting appears minor. Next, however, we will see empirically that as the dimension L grows, the benefit of the inflated argmax can be substantial.

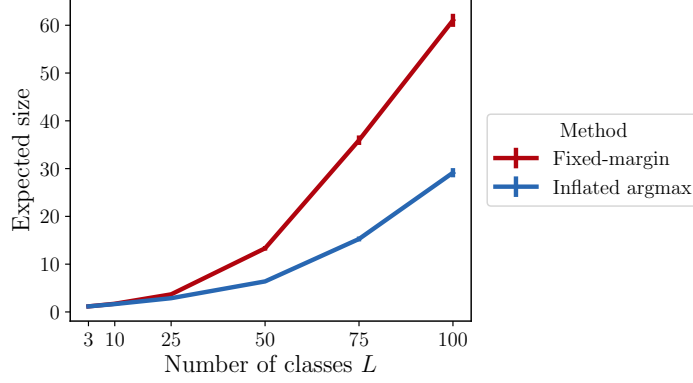


Figure 4: Simulation to compare the selection rules $\text{argmax}^\varepsilon$ and $s_{\text{margin}}^\varepsilon$. The figure shows the average set size, $|\text{argmax}^\varepsilon(w)|$ and $|s_{\text{margin}}^\varepsilon(w)|$, averaged over 1,000 random draws of w (with standard error bars shown). See Appendix D.2 for details.

D.2 Simulation to compare the selection rules

In this section, we compare the fixed-margin rule to the inflated argmax using randomly generated probability weights. For our simulation, we consider the size of the two sets for various values of L . To sample the weights, we draw a standard Gaussian random vector $Z \sim \mathcal{N}(0, \mathbf{I}_L)$ and define w as the softmax of Z , i.e.,

$$w_j = \frac{e^{Z_j}}{\sum_{\ell \in [L]} e^{Z_\ell}}$$

for $j \in [L]$. We then compute $\text{argmax}^\varepsilon(w)$ and $s_{\text{margin}}^\varepsilon(w)$, where $\varepsilon = 0.1$. In Figure 4, we present results comparing the average size of the sets returned by each of the two methods. In this setting, we see that the inflated argmax has substantial gains over the fixed-margin selection rule when the number of classes L is large. Even when $L = 25$, the ratio of the expected sizes is about 78%, so the inflated argmax has a nontrivial advantage in that setting. When $L = 100$, the ratio of expected sizes is 48%, meaning the inflated argmax is on average more than twice as small.

D.3 Proofs for the fixed-margin selection rule

D.3.1 Proof of Proposition 19

Let $w, v \in \mathbb{R}^L$ with $s_{\text{margin}}^\varepsilon(w) \cap s_{\text{margin}}^\varepsilon(v) = \emptyset$. Let $j \in \text{argmax}_\ell w_\ell$ and $k \in \text{argmax}_\ell v_\ell$. Then clearly $j \in s_{\text{margin}}^\varepsilon(w)$, so since $s_{\text{margin}}^\varepsilon(w) \cap s_{\text{margin}}^\varepsilon(v) = \emptyset$, this implies $j \notin s_{\text{margin}}^\varepsilon(v)$. By definition, then, $v_j \leq \max_\ell v_\ell - \varepsilon/\sqrt{2} = v_k - \varepsilon/\sqrt{2}$. By an identical argument, we have $w_k \leq w_j - \varepsilon/\sqrt{2}$. Then

$$\begin{aligned} \|w - v\|^2 &\geq (w_j - v_j)^2 + (w_k - v_k)^2 \\ &\geq (w_j - v_j)_+^2 + (v_k - w_k)_+^2 \\ &\geq \left(w_j - (v_k - \varepsilon/\sqrt{2})\right)_+^2 + \left(v_k - (w_j - \varepsilon/\sqrt{2})\right)_+^2 \\ &\geq \inf_{t \in \mathbb{R}} \left\{ (\varepsilon/\sqrt{2} - t)_+^2 + (\varepsilon/\sqrt{2} + t)_+^2 \right\} \\ &= 2(\varepsilon/\sqrt{2})^2 = \varepsilon^2. \end{aligned}$$

This proves $\|w - v\| \geq \varepsilon$, and therefore, we have shown that ε -compatibility is satisfied.

D.3.2 Proof of Proposition 20

Suppose $j \in \operatorname{argmax}^\varepsilon(w)$. Then $\operatorname{dist}(w, R_j^\varepsilon) < \varepsilon$, so we can find some $v \in R_j^\varepsilon$ with $\|w - v\| < \varepsilon$. Then, for any $k \neq j$,

$$\begin{aligned}
w_j - w_k &= v_j - v_k + (w_j - v_j) + (v_k - w_k) \\
&\geq \varepsilon/\sqrt{2} + (w_j - v_j) + (v_k - w_k) \text{ since } v \in R_j^\varepsilon \\
&\geq \varepsilon/\sqrt{2} - (|w_j - v_j| + |w_k - v_k|) \\
&\geq \varepsilon/\sqrt{2} - \sqrt{2}\sqrt{(w_j - v_j)^2 + (w_k - v_k)^2} \\
&\geq \varepsilon/\sqrt{2} - \sqrt{2}\|w - v\| \\
&> \varepsilon/\sqrt{2} - \sqrt{2} \cdot \varepsilon \\
&= -\varepsilon/\sqrt{2}.
\end{aligned}$$

Since this holds for all $k \neq j$, then,

$$w_j > \max_{k \in [L]} w_k - \varepsilon/\sqrt{2},$$

which proves that $j \in s_{\operatorname{margin}}^\varepsilon(w)$.

E Additional experimental results

In this section, we extend our experiment from Section 4 to consider some additional existing methods as baselines and evaluate each method using some common metrics from set-valued classification.

Selection rules. We compare the following selection rules:

1. Standard argmax .
2. ε -inflated argmax with tolerance $\varepsilon = .05$.
3. Top- K classification with $K = 2$.
4. Thresholding: including classes in the output set until the sum of probabilities becomes at least $\tau = 0.8$, i.e.,

$$\begin{aligned}
\Gamma_\tau^*(w) &:= \left\{ \ell \in [L] : w_\ell \geq w_{[\hat{k}]} \right\}, \text{ where} \\
\hat{k} &= \min\{k : w_{[1]} + \dots + w_{[k]} \geq \tau\}.
\end{aligned}$$

5. Nondeterministic classification optimized for F_1 -score [DDB09]:

$$\begin{aligned}
\operatorname{NDC}_{F_1}(w) &:= \left\{ \ell \in [L] : w_\ell \geq w_{[\hat{k}]} \right\}, \text{ where} \\
\hat{k} &= \min\{k : w_{[1]} + \dots + w_{[k]} \geq (k+1)w_{[k+1]}\},
\end{aligned}$$

with the convention $w_{[L+1]} = 0$.

6. Set-valued Bayes-optimal prediction [MWDH21]: for any utility $u : [L] \times 2^{[L]} \setminus \{\emptyset\} \rightarrow \mathbb{R}_+$,

$$\operatorname{SVBOP}_u(w) := \operatorname{argmax}_{S \subseteq [L]} \sum_{\ell \in [L]} w_\ell \cdot u(\ell, S).$$

We consider two utility functions based on u_{65} and u_{80} , defined below.

Evaluation metrics. We evaluate each method based on a variety of metrics. In addition to $\beta_{\operatorname{prec}}$ and $\beta_{\operatorname{set-size}}$, defined in Section 4, we assess each method in terms of *utility-discounted predictive accuracy* [ZCM12]. For a set $S \subseteq [L]$ and label $\ell \in [L]$, define for some parameters $\alpha, \beta > 0$,

$$u(\ell, S) := \mathbf{1}\{\ell \in S\} \cdot \left(\frac{\alpha}{|S|} - \frac{\beta}{|S|^2} \right).$$

Sel. rule	Algo.	$\beta_{\text{correct-single}} \nearrow$	$\beta_{\text{set-size}} \searrow$	$u_{65} \nearrow$	$u_{80} \nearrow$	$\beta_{\text{sup. infl.}} \searrow$
argmax	\mathcal{A}	0.879 (0.003)	1.000 (0.000)	0.879 (0.003)	0.879 (0.003)	-
argmax	$\tilde{\mathcal{A}}_m$	0.892 (0.003)	1.000 (0.000)	0.892 (0.003)	0.893 (0.003)	-
argmax ^e	\mathcal{A}	0.873 (0.003)	1.015 (0.001)	0.881 (0.003)	0.883 (0.003)	0.496 (0.005)
argmax ^e	$\tilde{\mathcal{A}}_m$	0.886 (0.003)	1.017 (0.001)	0.895 (0.003)	0.897 (0.003)	0.474 (0.005)
top-2	\mathcal{A}	0.000 (0.000)	2.000 (0.000)	0.632 (0.001)	0.777 (0.001)	0.905 (0.003)
top-2	$\tilde{\mathcal{A}}_m$	0.000 (0.000)	2.000 (0.000)	0.632 (0.001)	0.777 (0.001)	0.918 (0.003)
$\Gamma_{0.8}^*$	\mathcal{A}	0.756 (0.004)	1.248 (0.005)	0.880 (0.002)	0.909 (0.002)	0.622 (0.005)
$\Gamma_{0.8}^*$	$\tilde{\mathcal{A}}_m$	0.704 (0.005)	1.356 (0.006)	0.867 (0.002)	0.907 (0.002)	0.701 (0.005)
NDC for F_1	\mathcal{A}	0.832 (0.004)	1.105 (0.003)	0.889 (0.003)	0.902 (0.003)	0.531 (0.005)
NDC for F_1	$\tilde{\mathcal{A}}_m$	0.825 (0.004)	1.138 (0.004)	0.898 (0.003)	0.915 (0.002)	0.586 (0.005)
SVBOP _{u_{65}}	\mathcal{A}	0.838 (0.004)	1.091 (0.003)	0.889 (0.003)	0.901 (0.003)	0.525 (0.005)
SVBOP _{u_{65}}	$\tilde{\mathcal{A}}_m$	0.833 (0.004)	1.119 (0.003)	0.899 (0.003)	0.915 (0.002)	0.577 (0.005)
SVBOP _{u_{80}}	\mathcal{A}	0.777 (0.004)	1.190 (0.004)	0.885 (0.003)	0.910 (0.002)	0.611 (0.005)
SVBOP _{u_{80}}	$\tilde{\mathcal{A}}_m$	0.744 (0.004)	1.245 (0.005)	0.882 (0.002)	0.914 (0.002)	0.690 (0.005)

Table 2: Results on the Fashion MNIST data set. The table displays the frequency of returning the correct label as a singleton $\beta_{\text{correct-single}}$, average size $\beta_{\text{set-size}}$, utility-discounted predictive accuracies u_{65} and u_{80} , and the superfluous inflation, $\beta_{\text{sup. infl.}}$. For each metric, the symbol \nearrow indicates that higher values are desirable, while \searrow indicates that lower values are desirable. Results for the base algorithm are in white, and results for the subbagged algorithm are in gray. Standard errors are in parentheses.

We use the measures u_{65} with $(\alpha, \beta) = (1.6, 0.6)$ and u_{80} with $(\alpha, \beta) = (2.2, 1.2)$, which respectively give small and large rewards for being cautious [NDMH18].

Finally, we directly assess the extent to which each selection rule resorts to returning a non-singleton set on difficult instances. Specifically, we define the *superfluous inflation*, which, for a selection rule s , is the ratio of the accuracy of the standard argmax given s returns at least two labels divided by the accuracy of s given s returns at least two labels:

$$\beta_{\text{sup. infl.}} := \frac{\sum_{j=1}^N \mathbf{1} \left\{ \tilde{Y}_j \in \text{argmax}(\hat{p}(\tilde{X}_j)), |s(\hat{p}(\tilde{X}_j))| \geq 2 \right\}}{\sum_{j=1}^N \mathbf{1} \left\{ \tilde{Y}_j \in s(\hat{p}(\tilde{X}_j)), |s(\hat{p}(\tilde{X}_j))| \geq 2 \right\}}.$$

If this ratio is close to 1, it means the standard argmax is correct as often as s (when the latter expresses uncertainty by returning multiple labels), so outputting multiple labels could be seen as overly conservative. Note that the argmax never returns more than a singleton, so its superfluous inflation is left blank in our results.

Results. This experiment shows that the inflated argmax combined with the subbagged algorithm $\tilde{\mathcal{A}}_m$ has accuracy (according to a variety of metrics) comparable with several alternative methods, does not output overly large sets of candidate labels, and at the same time admits rigorous stability guarantees. That is, our algorithmic framework does not unduly harm empirical performance.

In Table 2, we present results for each selection rule applied to the base algorithm \mathcal{A} and the subbagged algorithm $\tilde{\mathcal{A}}_m$ described in Section 4. The inflated argmax has significantly higher average precision and significantly smaller set sizes than all of the alternative set-valued classifiers. These two measures are related, since a selection rule can only have high precision if it frequently returns the correct label *as a singleton set*. The inflated argmax also has the lowest superfluous inflation, meaning that it tends return multiple labels on difficult instances. The u_{65} of the inflated argmax is also within two standard errors of the u_{65} of SVBOP₆₅, which seeks to optimize this utility. Our method does have a significantly lower u_{80} than many of the competing methods, since this utility is more forgiving of returning multiple labels.

While this experiment considers many different perspectives on set-valued classification, we reiterate that our chief contribution is distribution-free stability guarantees. This means that, regardless of the dataset or base algorithm used, we can guarantee that our method will be stable. In the context of our experiment, Theorem 6 guarantees that $\delta_j \leq \delta^* = \varepsilon^{-2} \cdot \frac{1-1/L}{n-1} \cdot \frac{p_{n,m}}{1-p_{n,m}} \approx 0.006$ for *every* test point $j = 1, \dots, N$. Furthermore, our optimality result shows that the inflated argmax returns a singleton as often as possible among all ε -compatible selection rules.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and Section 1.2 in the introduction clearly state our contributions.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss limitations of the work in Section 5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: All the theorems are numbered and cross-referenced in the supplemental material, where the proofs appear.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all necessary details to reproduce the analysis in Section 4. To reproduce the learning algorithm \mathcal{A} , we both provide a reference and release our own documented code using the same architecture.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We attach our code in our submission to OpenReview, and we will de-anonymize the link to the Github repository after the review process.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experimental setting is presented in the core of the paper with enough detail to appreciate the results. The full details are provided with the code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Results in Table 1 are reported with standard errors.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Section 4 we provide details on the type and quantity of compute workers and total time for the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have reviewed the Code of Ethics and our work conforms to it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our research contribution is primarily theoretical and does not have immediately foreseeable impacts on society on a large scale.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: PyTorch and Fashion MNIST are properly credited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.