Coresets For Monotonic Functions with Applications to Deep Learning

Elad Tolochinsky, Dan Feldman

February 20, 2018

Abstract

Coreset (or core-set) in this paper is a small weighted *subset* Q of the input set P with respect to a given *monotonic* function $f : \mathbb{R} \to \mathbb{R}$ that *provably* approximates its fitting loss $\sum_{p \in P} f(p \cdot x)$ to *any* given $x \in \mathbb{R}^d$. Using Q we can obtain approximation to x^* that minimizes this loss, by running *existing* optimization algorithms on Q. We provide: (i) a lower bound that proves that there are sets with no coresets smaller than $n = |P|$, (ii) a proof that a small coreset of size near-logarithmic in n exists for *any* input P, under natural assumption that holds e.g. for logistic regression and the sigmoid activation function. (iii) a generic algorithm that computes Q in $O(nd+n \log n)$ expected time, (iv) novel technique for improving existing deep networks using such coresets, (v) extensive experimental results with open code.

1 Motivation

Traditional algorithms in computer science and machine learning are usually tailored to handle only off-line finite data set that is stored in memory. However, many modern systems do not use this computation model.

For example, GPS data from millions of smartphones, high definition images, YouTube videos, Twitter's text twitts, or audio signals from music or speech recognition arrive in a streaming fashion. The era of Internet of Things (IoT) provides us with wearable devices and mini-computers that collect data sets that are being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), genome sequencing, cameras, microphones, radio-frequency identification chips, finance (such as stocks) logs, internet search, and wireless sensor networks [17, 23, 11].

Limited memory. In such systems the input is an infinite stream that may be grown in practive to peta bytes of data-sets, and cannot be stored in memory. The data may arrive in real-time, and not just being read from a hard drive, so only one-pass over data and small memory is allowed.

Parallel computations. Even if we have streaming algorithms to maintain and learn Big data in memory from million of users, it is not reasonable to apply them on our laptop, and a large set of computation machines is used instead. However, using, for example, GPUs that run thousands of threads in parallel require us to design parallel version of our algorithms, which may be very hard to design and debug.

Distributed computations. If the data-set is distributed among many machines, e.g. network or "cloud", there is an additional problem of non-shared memory, which may be replaced by expensive and slow communication between the machines.

Limited computation power. Modern computation devices such as GPUs pose additional challenges since in order to run efficiently in parallel, unlike CPUs, only limited set of simple commands and algorithms may be used. However, unlike modern GPU cards that are plugged into expensive and strong servers on

the cloud, IoT devices are usually small and low cost. This results in a very weak computation power that is similar to computers in the previous century, as well as energy (battery) consuming issues that avoid us from running CPU extensive algorithms.

Weak or no theoretical guarantees. Due to the modern computation models above, learning even trivial properties of the data can become a non trivial task, as stated in [11]. These problems are felt especially within the scope of machine learning applications, where the common optimization functions and and model may be NP-hard to compute, already in the off-line settings. The result is neglecting, in some sense, decades of theoretical computer science research, and replacing it by fast heuristics and ad-hoc rules that have no theoretical guarantees but may be easy to implement, with reasonable results. Sometimes the papers include proofs of weak guarantees such as fast running time (with no approximation guarantee), convergence to a local minima (but not global, and in unbounded amount of time), or somewhat unnatural assumptions regarding the input or the behaviour of the algorithms.

Deep learning suggests to minimize a function that can be defined in what is known as a neural network. While it is considered a serious breakthrough in AI, and the state-of-the-art in many practical applications, it has very little theoretical guarantees regarding the relation between the output parameters (network) and the optimal network. This is not surprising as, it is known that even computing the optimal parameters of a single layer with a single neuron that uses the sigmoid activation function is NP-hard, including reasonable approximations [24]. Modern deep networks may consists of hundreds of layers, each with dozens of neurons with many different functions.

2 Coresets

Coresets suggest a natural solution or at least a very generic approach to attack the above challenges without re-inventing computer science, have some promising theoretical guarantees, and still use the success of existing heuristics. The idea is that instead of suggesting a new algorithm to solve the problem at hand from scratch, we summarize the data and reduced it in some sense, so that we can compute the optimal solution on the coreset using *existing* algorithms, while still getting provable approximation. The main challenge is to prove that there is a good trade-off between the coreset size and the guaranteed approximation. The exact coreset definition and its guarantees is inconsistent, as well as the name of the new set. Hence, it makes more sense to estimate the quality of a coreset by its properties, such as the following two properties.

Composable coresets refer to the output of coreset constructions that can be computed independently on different machines for different data-sets P_1 and P_2 to obtain the coresets C_1 and C_2, then be merged to their union $C_1 \cup C_2$, and re-compressed to a coreset C_3 of $C_1 \cup C_2$. If the "coreset for coresets" C_3 is a coreset for the union of the original sets $P_1 \cup P_2$, then the coreset construction outputs *composable coresets*. Unlike othe type of coresets, composable coreset allow us to handle Big data as follows.

Streaming, and distributed updates of the data using small memory and update time per point can be obtained from *any* (off-line) composable coreset scheme that outputs a coreset of a small size. We can also compute such coresets on distributed data (e.g. in cloud or smartphones), or dynamic data (with point deletion support in near-logarithmic time, but linear memory, e.g hard drive). This is now a common technique, known as *merge-and-reduce tree* and is explained in details in many papers; see e.g. [11] and references therein. Such coresets can be computed also on data that is distributed and streamed simultaneously as was proved in [14].

Based on this classic reduction, for the rest of the paper we focus only on off-line (but composable) coreset construction.

Weighted subset that is also a coreset, means that the coreset is essentially a small subset of the input points. Each point in the coreset is also associated with a positive (real) weight. Intuitively, the weight of a coreset point tells us how many points it represents in the original data. Indeed, the sum of weights in the coreset is usually approximately the size n of the input set. Weighted coresets has many advantages over other type of coresets, such as e.g. linear combinations of points, sometimes called sketches. For example, (i) Generalizing existing algorithms to handle weighted input points of the coreset is usually easy or exist (as the public code we used in this paper), (ii) If the input is sparse, then the coreset is also sparse, (iii) interpretation of the coreset is easier, (iv) numerical errors are usually small compared to, e.g., linear combinations of points when positive and negative coefficient cancel themselves in theory but not in practice.

Coresets as a bridge between theory and heuristics. In theory we should run the optimal algorithm on the coreset to get an approximation for the optimal solution of the real data. In practice, as stated in the previous section, for many of the problems in machine learning (such as deep learning) we do not have such provable optimal solution or even non-trivial approximations. Instead we run our favorite existing heuristic on the coreset. Since the coreset is small, we can run these heurisitcs many times on the coreset instead of one time on the full original data. Due do this reason, and also since coreset removes noise and smooth the optimization function in some sense, we usually get *better* result (i.e., negative ε) in practice by running the heuristic many times (e.g. from different initial seeds) on the coreset. Indeed, this is the case in this paper when we run our coreset on heuristics for optimizing the sigmoid function over the input.

3 Our contribution

We assume that we are given a set P of n points in $\mathbb{R}^d$, and a non-decreasing monotonic functions $f : \mathbb{R} \to \mathbb{R}$. Such a function represents a loss of fitting kernel function (model, classifiers). For example, $f(y) = \frac{1}{1+e^{-y}}$. for the case of sigmoid function, and $f(y) = \ln(1 + e^y)$ for the case of logistic regression, which are both used as activation functions in the last layer of neural networks for obtaining the final classification (probability between 0 and 1) for each label class. The total loss or sum of errors for every $x \in \mathbb{R}^d$ is then $\sum_{p \in P} f(p \cdot x)$, where p may be multiplied by its label $y \in \{0, 1\}$ for supervised data.

For a given error parameter $\varepsilon \in (0, 1)$, we wish to compute an *ε-coreset* $Q \subseteq P$, with a weight function $u : Q \to [0, \infty)$ that *provably* approximates the fitting cost of P for every $x \in \mathbb{R}^d$, up to a multiplicative factor of $1 \pm \varepsilon$, i.e.,

$$(1 - \varepsilon) \sum_{p \in P} f(p \cdot x) \leq \sum_{p \in Q} w(p) f(p \cdot x) \leq (1 + \varepsilon) \sum_{p \in P} f(p \cdot x).$$

Our results are as follows. (i) A lower bound that proves that there are no such small coresets in general. More precisely, there are input sets with no coresets of size smaller than n, for every given $\varepsilon \in (0, 1)$ and integer $n \geq 1$.

(ii) To overcome this lower bound, we add natural assumptions regarding f, mainly a regularization term to the loss function, which is often added anyway to avoid overfitting. In fact, without this term the function is minimizes where x approaches infinity. However, after adding the regularization term, the sigmoid function above becomes $g(p, x) = f(p, x) + \|x\|_2^2 / k$, where $k > 0$ defines the trade-off between minimizing the function and the complexity (length, in this case) of the set of parameters.

While minimizing such functions may still be NP-hard (such as in the sigmoid case), we prove that a coreset Q of size that is near-logarithmic in n exists for *any* input set P.
(iii) A generic algorithm that computes the coreset Q above in $O(nd + n \log n)$ expected time. Unlike most existing algorithms and results, the algorithm bounds sensitivity for general sets of monotonoic functions, and not a specific function. We can then obtain approximation to the desired model of P by running *existing* algorithms on Q, that can be computed for streaming and distributed Big data.

(iv) Novel technique for applying our coresets to deep learning in order to get *better* classifiers than the state of the art.

(v) An open-code implementation of our algorithm [1], and extensive experimental results on both synthetic and real-world public datasets.

4 Related Work

In [15] Har-Peled shows how to construct a coreset of one dimensional points sets ($d = 1$) for sums of single variable real valued functions. In the scope of machine learning most of the research involves clustering techniques [12, 13, 10] and regressions [2, 6, 30]. Several coresets were constructed for supervised learning problems including coresets for Gaussian mixture models [8], and SVM [25, 16].

The work by [18] introduces lower bounds on the total sensitivity of the logistic regression problem that is used in this paper. It also introduces an upper bound for the total sensitivity and coreset size based on k-clustering coresets. However the bounds hold only for input set P from very specific distributions (roughly, when P is well separated into k clusters).

The main tool of this work uses the unified framework presented in [9], which was recently improved in [3]. We also use the reduction from $\mathcal{L}_\infty$ coresets that approximates $\max_{p\in P} f(p \cdot x)$ to our $\mathcal{L}_1$ coreset (sum of loss) which was introduced in [27].

5 Overview

Our algorithm is based on previous results that are summarized in Section 6. Mainly, the fact that in order to compute a coreset (which is a weighted subset) for a loss function it suffices to bound the sensitivity (importance) of each point and the VC-dimension of the related function, as defined in the section. The size of the coreset depends on the sum of sensitivities over all the points, the VC-dimension, and the desired approximation error ε. A bound on the VC-dimension for the family of monotonic function is known to be $O(d)$ [18], so the majority of the paper is devoted to bound the sensitivity of each point.

In Section 7 we show example input sets that have no coreset that is smaller than the input size, for monotonic functions. This motivates the necessity of the assumptions in Section 8 regarding the properties of the function. Mainly, that it includes a regularization term that depends on $\|x\|$. This term is usually added anyway, both in theory and practice, to reduce the complexity of the model and avoid overfitting, where $k > 0$ determines the tradeoff between minimizing $\sum_{p\in P} f(p, x)$ and using very large x. In fact, without this term k, the trivial minimizer is usually $x = \infty$. In Section 8 we also introduce our main generic algorithm for coreset construction for such families of monotonic functions. After stating the general result, we demonstrate it for a coreset for the sigmoid activation function.

In Section 11 we show experimental results on synthetic and real data sets. In particular, we show a technique to improve the fitting cost of existing neural network by computing coreset for the input to its last layer, and update its weights.

6 Preliminaries

We first describe the framework of [9] for computing coresets for certain optimization problems. The framework is based on a non-uniform sampling technique. We sample points with different probabilities in such a way that points that have a high influence on the optimization problem are sampled with higher probability, to make sure that the sample contains the important points. At the same time, in order to keep the sample unbiased, the sample points are weighted reciprocal to their sampling probability. To quantify the influence of single point on the optimization problem, Feldman and Langberg uses a term that was named *sensitivity* in [21].

Definition 1 (Query space)**.** *Let P be a finite set called* points*, and $w : P' \to (0, \infty)$ for some $P' \supseteq P$ be called a* weight function*. Then (P, w) is called a* weighted set*. A special case is $(P, \mathbf{1})$ where $w(p) = 1$ for*

every $p \in P$. Let X be a set called queries*, and $c : P \times X \to [0, \infty)$ be a given* cost *or* loss *function. The* total cost *of P with respect to a query $\boldsymbol{x} \in X$ is*

$$C(P, w, \boldsymbol{x}) = \sum_{\boldsymbol{p} \in P} w(\boldsymbol{p})\, c(\boldsymbol{p}, \boldsymbol{x}).$$

The tuple (P, w, X, c) is called a query space.

Definition 2 (Sensitivity)**.** *[9, 21] The* sensitivity *of a point $\boldsymbol{p} \in P$ in a query space (P, w, X, c) is*

$$s(p) = s_{P,w,X,c}(\boldsymbol{p}) = \sup_{\boldsymbol{x} \in X} \frac{w(\boldsymbol{p})\, c(\boldsymbol{p}, \boldsymbol{x})}{C(P, w, \boldsymbol{x})},$$

where the supermum is over every $\mathbf{x} \in X$ such that $C(P, w, \boldsymbol{x}) > 0$. The total sensitivity *of the query space is $t(P) = t(P, w, X, c) = \sum_{\boldsymbol{p} \in P} s(\boldsymbol{p})$.*

The main contribution of Feldman and Langberg is to establish a connection to the theory of range spaces and VC-dimension. The dimension of a query space is a measure to its combinatorial complexity.

Definition 3 (VC-dimension)**.** *[9, 26] For a query space (P, w, X, c) we define*

$$range(\boldsymbol{x}, r) = \{\boldsymbol{p} \in P \mid w(\boldsymbol{p})\, c(\boldsymbol{p}.\boldsymbol{x}) \leq r\},$$

for every $x \in X$ and $r \geq 0$. The dimension of (P, w, X, c) is the size $|G|$ of the largest subset $G \subseteq P$ such that have

$$|\{G \cap range(\boldsymbol{x}, r) \,|\boldsymbol{x} \in X, r \geq 0\}| = 2^{|G|}.$$

Feldman and Langberg show how to compute a weighted subset (Q, u) that will approximate the total cost $C(P, w, \boldsymbol{x})$ for every query, up to a multiplicative factor of $1 \pm \varepsilon$ without further assumptions. Such a set is sometimes called a *coreset* as follows,

Definition 4 (ε-coreset)**.** *Let (P, w, X, c) be a query space, and $\varepsilon \in (0, 1)$ be an error parameter. An* ε-coreset of (P, w, X, c) *is a weighted set (Q, u) such that*

$$\forall \boldsymbol{x} \in X : |C(P, w, \boldsymbol{x}) - C(Q, u, \boldsymbol{x})| \leq \varepsilon C(P, w, \boldsymbol{x}).$$

In [9] it was proved how small total sensitivity implies small coreset, and the size was reduced lately in [3].

Theorem 5 (coreset construction)**.** *[3, 9] Let (P, w, X, c) be a query space of dimension d and total sensitivity t. Let $\varepsilon, \delta \in (0, 1)$. Let Q be a random sample of*

$$|Q| \geq \frac{10t}{\varepsilon^2}\left(d \log t + \log\left(\frac{1}{\delta}\right)\right),$$

i.i.d points from P, such that for every $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$ we have $\boldsymbol{p} = \boldsymbol{q}$ with probability $\frac{1}{t} \cdot s_{P,w,X,c}(\boldsymbol{p})$. Let $u(\boldsymbol{p}) = \frac{tw(\boldsymbol{p})}{s_{P,w,X,c}(\boldsymbol{p})|Q|}$ for every $\boldsymbol{p} \in Q$. Then, with probability at least $1 - \delta$, (Q, u) is an ε-coreset of (P, w, X, c).

7 Lower Bounds

In this section we show that without adding additional assumption on the function, no coreset exist for monotonic function f that satisfies

$$\lim_{x \to \infty} \frac{f(-x)}{f(x)} = 0. \tag{1}$$

That is, for every $n \geq 1$, we can find a set P of size n such that any corest of P is of size n. The reason we chose to focus on this property is because most of the common functions used for learning satisfy this property.

To see this we will use the notion of total sensitivity defined above. Theorem 5 states that a small upper bound on the sensitivity is a sufficient condition for the existence of a coreset. We will show that this is also a necessary condition in the sense that if the sensitivity of every point is too large, no non-trivial coreset can exist.

Lemma 6 (Lower bound via Total sensitivity)**.** *Let (P, w, X, c) be a query space, and $\varepsilon \in (0, 1)$. If every $\boldsymbol{p} \in P$ has sensitivity $s_{P,w,X,c}(\boldsymbol{p}) = 1$, then for every ε-coreset (Q, u) we have $Q = P$.*

Proof. Let (Q, u) be a weighted set, where $Q \subset P$. It suffices to prove that (Q, u) is not an ε-coreset for P. Denote

$$u_{\max} \in \arg\max_{\boldsymbol{p} \in Q} u(\boldsymbol{p}), \text{ and } w_{\min} \in \arg\min_{\boldsymbol{p} \in P} w(\boldsymbol{p}).$$

Let $p \in P \setminus Q$. By the assumption $s_{P,w,X,c}(\boldsymbol{p}) \geq 1$, there is $\boldsymbol{x_p} \in X$ such that

$$\frac{w(\boldsymbol{p})\, c(\boldsymbol{p}, \boldsymbol{x_p})}{C(P, w, \boldsymbol{x_p})} = 1 > \frac{u_{\max}}{u_{\max}} - \frac{w_{\min}(1-\varepsilon)}{u_{\max}}.$$

Multiplication by $C(P, w, \boldsymbol{x}_p)$ yields

$$\begin{aligned} &w(\boldsymbol{p})\, c(\boldsymbol{p}, \boldsymbol{x_p}) > \\ &\frac{u_{\max} - w_{\min}(1-\varepsilon)}{u_{\max}} \cdot C(P, w, \boldsymbol{x_p}). \end{aligned} \tag{2}$$

We have that

$$\begin{aligned} C(Q, u, \boldsymbol{x_p}) &= \sum_{\boldsymbol{q} \in Q} u(\boldsymbol{q})\, c(\boldsymbol{q}, \boldsymbol{x_p}) \\ &= \sum_{\boldsymbol{q} \in Q} \frac{u(\boldsymbol{q})}{w(\boldsymbol{q})} w(\boldsymbol{q})\, c(\boldsymbol{p}, \boldsymbol{x_p}) \leq \frac{u_{\max}}{w_{\min}} \sum_{\boldsymbol{q} \in Q} w(\boldsymbol{q})\, c(\boldsymbol{q}, \boldsymbol{x_p}) \\ &\leq \frac{u_{\max}}{w_{\min}} \sum_{\boldsymbol{p'} \in P \setminus \{\boldsymbol{p}\}} w(\boldsymbol{p})\, c(\boldsymbol{p'}, \boldsymbol{x_p}) \qquad (3) \\ &= \frac{u_{\max}}{w_{\min}} \left(C(P, w, \boldsymbol{x_p}) - w(\boldsymbol{p})\, c(\boldsymbol{p}, \boldsymbol{x_p}) \right) \\ &< \frac{u_{\max}}{w_{\min}} C(P, w, \boldsymbol{x_p}) \left(1 - \frac{u_{\max} - w_{\min}(1-\varepsilon)}{u_{\max}} \right) \qquad (4) \\ &= (1-\varepsilon)\, C(P, w, \boldsymbol{x_p}), \end{aligned}$$

where (3) is by the assumption $\boldsymbol{p} \in P \setminus Q$, and (4) is by (2). Hence Q cannot be used to approximate $C(P, w, \boldsymbol{x_p})$ and thus is not an ε-coreset for P. □

To complete the proof of our lower bound we now only need to show that there is a set of points for which the sensitivity of every point is 1. Together with the lemma above, this will complete the proof. The idea behind finding a set for which every point has sensitivity 1 is to find a set of points in which every point is linearly separable from the rest of the set. Such a set was shown to exist in [18].

Lemma 7. *[18] There is a finite set of points $P \subseteq \mathbb{R}^d$ such that for every $\boldsymbol{p} \in P$ and $R > 0$ there is $\boldsymbol{y_p} \in \mathbb{R}^d$ of length $\|\boldsymbol{y_p}\| \leq R$ such that $\boldsymbol{y_p} \cdot \boldsymbol{p} = -R$, and for every $\boldsymbol{q} \in P \setminus \{\boldsymbol{p}\}$ we have $\boldsymbol{y_p} \cdot \boldsymbol{q} \geq R$.*

We now prove that the sensitivity of every point in the set above is 1. We generalize a result from [18] by letting the cost be any function upholding the conditions of Theorem 8 and the data to be weighted.

Theorem 8. *Let $f : \mathbb{R} \to (0,\infty)$ be a non-decreasing monotonic function that satisfies* (1)*. and let $c(\boldsymbol{x},\boldsymbol{p}) = f(\boldsymbol{x}\cdot\boldsymbol{p})$ for every $\boldsymbol{x},\boldsymbol{p} \in \mathbb{R}^d$. Let $\varepsilon \in (0,1)$, $n \geq 1$ be an integer, and $w : \mathbb{R}^d \to (0,\infty)$. There is a set $P \subset \mathbb{R}^d$ of $|P| = n$ points such that if (Q,u) is an ε-coreset of $\left(P,w,\mathbb{R}^d,c\right)$ then $Q = P$.*

Proof. Let $P \subseteq \mathbb{R}^d$ be the set that is defined in Lemma 7, and let $\boldsymbol{p} \in P$, and $R > 0$. By Lemma 7, there is $\boldsymbol{y_p} \in \mathbb{R}^d$ such that $\boldsymbol{y_p}\cdot\boldsymbol{p} = -R$, and for every $\boldsymbol{q} \in P \setminus \{\boldsymbol{p}\}$ we have $-\boldsymbol{y_p}\cdot\boldsymbol{q} \leq -R$. By this pair of properties,

$$f(-\boldsymbol{y_p}\cdot\boldsymbol{p}) = f(R) \text{ and } f(-\boldsymbol{y_p}\cdot\boldsymbol{q}) \leq f(-R),$$

where in the last inequality we use the assumption that f is non-decreasing. By letting $\boldsymbol{x_p} = -\boldsymbol{y_p}$, we have

$$\frac{w(q)f(\boldsymbol{x_p}\cdot\boldsymbol{q})}{w(p)f(\boldsymbol{x_p}\cdot\boldsymbol{p})} = \frac{w(q)f(-\boldsymbol{y_p}\cdot\boldsymbol{q})}{w(p)f(-\boldsymbol{y_p}\cdot\boldsymbol{p})} \leq \frac{w(q)f(-R)}{w(p)f(R)}.$$

Therefore, by letting $w_{\max} \in \arg\max_{\boldsymbol{p}\in P} w(\boldsymbol{p})$,

$$\begin{aligned}
s_{P,w,\mathbb{R}^d,c}(\boldsymbol{p}) &\geq \frac{w(\boldsymbol{p})f(\boldsymbol{x_p}u\cdot\boldsymbol{p})}{\sum_{\boldsymbol{q}\in P} w(\boldsymbol{q})f(\boldsymbol{x_p}\cdot\boldsymbol{q})} \\
&= \frac{w(\boldsymbol{p})f(\boldsymbol{x_p}\cdot\boldsymbol{p})}{w(\boldsymbol{p})f(\boldsymbol{p}\cdot\boldsymbol{x_p}) + \sum_{\boldsymbol{q}\in P\setminus\{\boldsymbol{p}\}} w(\boldsymbol{q})f(\boldsymbol{x_p}\cdot\boldsymbol{q})} \\
&= \frac{1}{1+\sum_{\boldsymbol{q}\in P\setminus\{\boldsymbol{p}\}} \frac{w(\boldsymbol{q})f(\boldsymbol{x_p}\cdot\boldsymbol{q})}{w(\boldsymbol{p})f(\boldsymbol{x_p}\cdot\boldsymbol{p})}} \geq \frac{1}{1+\sum_{q\in P\setminus\{\boldsymbol{p}\}} \frac{w(\boldsymbol{q})f(-R)}{w(\boldsymbol{p})f(R)}} \\
&\geq \frac{1}{1+(n-1)\frac{w_{\max}f(-R)}{w(\boldsymbol{p})f(R)}}.
\end{aligned}$$

By replacing x with R in (1), we have

$$\lim_{R\to\infty} \frac{w_{\max}f(-R)}{w(\boldsymbol{p})f(R)} = \frac{w_{\max}}{w(\boldsymbol{p})} \lim_{R\to\infty} \frac{f(-R)}{f(R)} = 0.$$

Thus we obtain

$$s_{P,w,\mathbb{R}^d,c}(\boldsymbol{p}) = \sup_{R>0} \frac{1}{1+(n-1)\frac{w_{\max}f(-R)}{w(\boldsymbol{p})f(R)}} = 1.$$

Theorem 8 then follows from the last equality and Lemma 6. □

8 Coresets For Monotonic Bounded Functions

Lemma 9. *Let $P \subset \mathbb{R}^d$ be a finite set, $M > 0$, $f : \mathbb{R} \to (0,M]$ be a non-decreasing function. Let $g : [0,\infty) \to [0,\infty)$ and $k > 0$. For every $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{p'} \in P$ define $c_k(\boldsymbol{p'},\boldsymbol{x}) = f(\boldsymbol{p'}\cdot\boldsymbol{x}) + \frac{g(\|\boldsymbol{x}\|)}{k}$. Let $\boldsymbol{p} \in P$ and $b_{\boldsymbol{p}} > 0$ such that for every $z > 0$*

$$f(\|\boldsymbol{p}\|z) + \frac{g(z)}{k} \leq b_{\boldsymbol{p}}\left(f(-\|\boldsymbol{p}\|z) + \frac{g(z)}{k}\right). \tag{5}$$

Then for every $\boldsymbol{x} \in \mathbb{R}^d$

$$\max_{\boldsymbol{p'}\in P} c_k(\boldsymbol{p'},\boldsymbol{x}) \leq \frac{M}{f(0)}(b_{\boldsymbol{p}}+1)\,c_k(\boldsymbol{p},\boldsymbol{x}).$$

Proof. Let $\boldsymbol{x} \in \mathbb{R}^d$ and $q \in P$ such that $\boldsymbol{x}\cdot\boldsymbol{q} > 0$. We have, by the monotonic properties of f,

$$f(0) \leq f(\boldsymbol{x}\cdot\boldsymbol{q}). \tag{6}$$

Algorithm 1 MONOTONIC-CORESET$(P, \varepsilon, \delta, k)$

Input: A set P of n points in $\mathbb{R}^d$,
an error parameter $\varepsilon \in (0,1)$,
probability of failure $\delta \in (0,1)$, and
a real valued regularization term $k > 0$.
Output: An ε-coreset (Q, u) for $(P, \mathbf{1}, \mathbb{R}^d, c_{\text{sigmoid},k})$.

1: Sort the points in $P = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n\}$ by their length, i.e., $\|\boldsymbol{p}_1\| \leq \cdots \leq \|\boldsymbol{p}_n\|$.
2: **for** every $j \in \{1, \cdots, n\}$ **do**
3: Set $s(\boldsymbol{p}_j) \leftarrow \dfrac{132\sqrt{k}\left\|\boldsymbol{p}_j\right\| + 2}{j}$
4: **end for**
5: Set $t \leftarrow \sum_{i=1}^{n} s(\boldsymbol{p}_i)$
6: Set $m \leftarrow \dfrac{10t}{\varepsilon^2}\left(d \ln t + \ln \dfrac{1}{\delta}\right)$
7: Pick a sample $Q \subseteq P$ of $|Q| \geq \min\{m, n\}$ i.i.d. points such that for every $\boldsymbol{q} \in Q$ and $\boldsymbol{p} \in P$ we have $\boldsymbol{p} = \boldsymbol{q}$ with probability $s(\boldsymbol{p})/t$.
8: **for** every $\boldsymbol{p}_i \in Q$ **do**
9: Set $u(\boldsymbol{p}_i) \leftarrow \dfrac{1}{|Q| \operatorname{Prob}(\boldsymbol{p}_i)}$
10: **end for**
11: **return** (Q, u)

Hence,

$$\max_{\boldsymbol{p}' \in P} f(\boldsymbol{x} \cdot \boldsymbol{p}') \leq M = \frac{M}{f(0)} f(0) \leq \frac{M}{f(0)} f(\boldsymbol{x} \cdot \boldsymbol{q}), \tag{7}$$

where the first inequality is since f is bounded by M, and the last inequality is by (6). By adding $\frac{g(\|\boldsymbol{x}\|)}{k}$ to both sides of (7) and since $1 \leq \frac{M}{f(0)}$ we obtain,

$$\begin{aligned} \max_{p' \in P} c_k(p', x) &= \max_{\boldsymbol{p}' \in P} f(\boldsymbol{x} \cdot \boldsymbol{p}') + \frac{g(\|\boldsymbol{x}\|)}{k} \\ &\leq \frac{M}{f(0)} f(\boldsymbol{x} \cdot \boldsymbol{q}) + \frac{g(\|\boldsymbol{x}\|)}{k} \\ &\leq \frac{M}{f(0)} \left(f(\boldsymbol{x} \cdot \boldsymbol{q}) + \frac{g(\|\boldsymbol{x}\|)}{k} \right). \end{aligned} \tag{8}$$

The rest of the proof follows by case analysis on the sign of $\boldsymbol{x} \cdot \boldsymbol{p}$, i.e. (i) $\boldsymbol{x} \cdot \boldsymbol{p} \geq 0$ and (ii) $\boldsymbol{x} \cdot \boldsymbol{p} < 0$.

Case (i): $\boldsymbol{x} \cdot \boldsymbol{p} \geq 0$. Substituting $q = p$ in (8) yields

$$\begin{aligned} \max_{p' \in P} c_k(p', x) &\leq \frac{M}{f(0)} \left(f(\boldsymbol{x} \cdot \boldsymbol{p}) + \frac{g(\|\boldsymbol{x}\|)}{k} \right) \\ &= \frac{M}{f(0)} c_k(p, x) \leq \frac{M}{f(0)} (b_p + 1) c_k(p, x), \end{aligned} \tag{9}$$

where the last inequality follows by the assumption $b_p > 0$. **Case (ii)**: $\boldsymbol{x} \cdot \boldsymbol{p} < 0$. In this case $\boldsymbol{x} \cdot (-\boldsymbol{p}) > 0$.

Substituting $q = -p$ in (8) yields

$$\max_{p' \in P} c_k(p', x) \leq \frac{M}{f(0)} \left(f(\boldsymbol{x} \cdot (-\boldsymbol{p})) + \frac{g(\|\boldsymbol{x}\|)}{k} \right) \tag{10}$$

$$\leq \frac{M}{f(0)} \left(f(\|\boldsymbol{x}\| \|\boldsymbol{p}\|) + \frac{g(\|\boldsymbol{x}\|)}{k} \right) \tag{11}$$

$$\leq \frac{M}{f(0)} b_{\boldsymbol{p}} \left(f(-\|\boldsymbol{x}\| \|\boldsymbol{p}\|) + \frac{g(\|\boldsymbol{x}\|)}{k} \right) \tag{12}$$

$$\leq \frac{M}{f(0)} b_{\boldsymbol{p}} \left(f(\boldsymbol{x} \cdot \boldsymbol{p}) + \frac{g(\|\boldsymbol{x}\|)}{k} \right), \tag{13}$$

$$= \frac{M}{f(0)} b_p c_k(p, x) \leq \frac{M}{f(0)} (b_p + 1) c_k(p, x), \tag{14}$$

where (11) and (13) are by the Cauchy-Schwartz inequality and the monotonicity of f, and (12) follows by substituting $z = \|x\|$ in (5). □

Theorem 10. *Let $\varepsilon, \delta \in (0,1)$, $\boldsymbol{p} \in P$ and $b_{\boldsymbol{p}} > 0$ such that for every $z > 0$*

$$f(\|\boldsymbol{p}\| z) + \frac{g(z)}{k} \leq b_{\boldsymbol{p}} \left(f(-\|\boldsymbol{p}\| z) + \frac{g(z)}{k} \right). \tag{15}$$

Then, there is a weighted set (Q, u) such that with probability at least $1 - \delta$, (Q, u) is an ε-coreset for $(P, \mathbf{1}, \mathbb{R}^d, c_k)$. Moreover, by letting $b_{\max} \in \arg\max_{\boldsymbol{p} \in P} b_{\boldsymbol{p}}$ and $t = (1 + \frac{M}{f(0)} b_{\max}) \ln n$,

$$|Q| \in O\left(\frac{t}{\varepsilon^2} \left(d \log t + \log \frac{1}{\delta} \right) \right)$$

Proof. Let $\boldsymbol{p} \in P$, by Lemma 9 we obtain

$$\max_{\boldsymbol{p'} \in P} c_k(\boldsymbol{p'}, \boldsymbol{x}) \leq \frac{M}{f(0)} (b_{\boldsymbol{p}} + 1) c_k(\boldsymbol{p}, \boldsymbol{x}) \leq \tag{16}$$

$$\frac{M}{f(0)} (b_{\max} + 1) c_k(\boldsymbol{p}, \boldsymbol{x}). \tag{17}$$

Where (16) is by Lemma 9 and (17) holds since for every $\boldsymbol{p} \in P$, $b_{\boldsymbol{p}} \leq b_{\max}$. Thus, $\{\boldsymbol{p}\}$ is an $\left[\left(\frac{M}{f(0)} (b_{\max} + 1) \right) - 1 \right]$-$\mathcal{L}_\infty$ coreset. Using the reduction in [27] we have that

$$t(P, \mathbf{1}, \mathbb{R}^d, c_k) \in O\left(t = (1 + \frac{M}{f(0)} b_{\max}) \ln n \right).$$

By 5 we obtain the required result. □

9 Example: Coreset For the Sigmoid Activation Function

We present an application to the framework described above for sums of sigmoid functions.

Lemma 11. *Let $f(z) = \frac{1}{1+e^{-z}}$ for every $z \in \mathbb{R}$ and let $c > 0$. There is $k_0 > 0$ such that for every $k \geq k_0$ and for every $z \geq 0$,*

$$\frac{f(cz) + \frac{z^2}{k}}{f(-cz) + \frac{z^2}{k}} \leq 66c\sqrt{k}.$$

Proof. See Lemma 20 in the appendix. □

Lemma 12. *Let $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subset \mathbb{R}^d$ be a set of points, sorted by their length. I.e. $\|\boldsymbol{p}_i\| \leq \|\boldsymbol{p}_j\|$ for every $1 \leq i \leq j \leq n$. Let $k > 0$ and $c_{\text{sigmoid},k}(\boldsymbol{p}, \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{p}\cdot\boldsymbol{x}}} + \frac{\|\boldsymbol{x}\|^2}{k}$ for every $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{p} \in P$. Then the sensitivity of every $p_j \in P$ is bounded by $s(\boldsymbol{p}) = s_{P,\mathbf{1},\mathbb{R}^d,c_{\text{sigmoid},k}}(\boldsymbol{p}) \in O\left(\frac{\|\boldsymbol{p}_j\|\sqrt{k}+1}{j}\right)$, and the total sensitivity is*

$$t = \sum_{\boldsymbol{p}\in P} s(\boldsymbol{p}) \in O\left(\log n + \sqrt{k}\sum_{j=1}^{n} \frac{\|\boldsymbol{p}_j\|}{j}\right).$$

Proof. Define $f(z) = \frac{1}{1+e^{-z}}$ and $g(z) = z^2$ for every $z \in \mathbb{R}$. Let $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{p}_j \in P$ and $i \in [1, j]$ be an integer. We substitute $c = \|\boldsymbol{p}_i\|$ in Lemma 19 to obtain that for every $z > 0$

$$\frac{f(\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}}{f(-\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}} \leq 66 \|\boldsymbol{p}_i\| \sqrt{k}.$$

Denote $b_{\boldsymbol{p}_i} = 66 \|\boldsymbol{p}_i\| \sqrt{k}$ and multiply the above term by $f(-\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}$ to get

$$f(\|\boldsymbol{p}_i\| z) + \frac{z^2}{k} \leq b_{\boldsymbol{p}_i}\left(f(-\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}\right).$$

Substituting in Lemma 9 $\boldsymbol{p} = \boldsymbol{p}_i, f(z) = \frac{1}{1+e^{-z}}, g(z) = z^2, M = 1, f(0) = \frac{1}{2}$ yields

$$\max_{\boldsymbol{p}'\in P} c_{\text{sigmoid},k}(\boldsymbol{p}', \boldsymbol{x}) \leq 2\left(b_{\boldsymbol{p}_i} + 1\right) c_{\text{sigmoid},k}(\boldsymbol{p}_i, \boldsymbol{x}). \tag{18}$$

Thus

$$\begin{aligned} c_{\text{sigmoid},k}\left(\boldsymbol{p}_j, \boldsymbol{x}\right) \leq & \max_{\boldsymbol{p}'\in P} c_{\text{sigmoid},k}(\boldsymbol{p}', \boldsymbol{x}) && (19)\\ \leq & 2\left(b_{\boldsymbol{p}_i} + 1\right) c_{\text{sigmoid},k}(\boldsymbol{p}_i, \boldsymbol{x}), && (20) \end{aligned}$$

where (26) is since $\boldsymbol{p}_j \in P$ and (27) is by (18). Dividing both sides by $2\left(b_{\boldsymbol{p}_i} + 1\right)$ yields

$$c_{\text{sigmoid},k}(\boldsymbol{p}_i, \boldsymbol{x}) \geq \frac{c_{\text{sigmoid},k}\left(\boldsymbol{p}_j, \boldsymbol{x}\right)}{2\left(b_{\boldsymbol{p}_i} + 1\right)}. \tag{21}$$

We now proceed to bound the sensitivity of $\boldsymbol{p}_j$. Since the set of points $\left\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_j\right\}$ is a subset of P, and since the cost function $c_{\text{sigmoid},k}\left(\boldsymbol{p}_j, \boldsymbol{x}\right)$ is positive we have that

$$\sum_{\boldsymbol{p}'\in P} c_{\text{sigmoid},k}(\boldsymbol{p}', \boldsymbol{x}) \geq \sum_{i=1}^{j} c_{\text{sigmoid},k}(\boldsymbol{p}_i, \boldsymbol{x}). \tag{22}$$

By summing (28) over $i \leq j$, we obtain

$$\begin{aligned} \sum_{i=1}^{j} c_{\text{sigmoid},k}(\boldsymbol{p}_i, \boldsymbol{x}) &\geq c_{\text{sigmoid},k}(\boldsymbol{p}_j, \boldsymbol{x}) \sum_{i=1}^{j} \frac{1}{2(b_{\boldsymbol{p}_i} + 1)} \\ &\geq c_{\text{sigmoid},k}(\boldsymbol{p}_j, \boldsymbol{x}) \frac{j}{2(b_{\boldsymbol{p}_j} + 1)}, \end{aligned} \tag{23}$$

where the last inequality holds since $b_{p_i} = 66 \|p_i\| \sqrt{k} \leq b_{\boldsymbol{p}_j}$ for every $i \leq j$. Combining (29) and (30) yields

$$\sum_{\boldsymbol{p}'\in P} c_{\text{sigmoid},k}(\boldsymbol{p}', \boldsymbol{x}) \geq \frac{j c_{\text{sigmoid},k}(\boldsymbol{p}_j, \boldsymbol{x})}{2(b_{\boldsymbol{p}_j} + 1)} \tag{24}$$

Therefore, the sensitivity is bounded by

$$s_{P,\mathbf{1},\mathbb{R}^d,c_{\text{sigmoid},k}}(\boldsymbol{p}_j) = \sup_{\boldsymbol{x}\in\mathbb{R}^d} \frac{c_{\text{sigmoid},k}(\boldsymbol{p}_j,\boldsymbol{x})}{\sum_{\boldsymbol{p'}\in P} c_{\text{sigmoid},k}(\boldsymbol{p'},\boldsymbol{x})} \leq \frac{2(b_{p_j}+1)}{j} \leq \frac{2(66\left\|p_j\right\|\sqrt{k}+1)}{j}.$$

Summing this sensitivity bounds the total sensitivity by

$$\sum_{j=1}^{n} \frac{2(66\left\|p_j\right\|\sqrt{k}+1)}{j} \in O\left(\log n + \sqrt{k}\sum_{j=1}^{n}\frac{\left\|p_j\right\|}{j}\right).$$

□

Theorem 13. *Let P be a set of n points in the unit ball of $\mathbb{R}^d$, $\varepsilon,\delta\in(0,1)$, and $k>0$. For every $p,x\in\mathbb{R}^d$, let*

$$c_{\text{sigmoid},k}(\boldsymbol{p},\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{p}\cdot\boldsymbol{x}}} + \frac{\left\|\boldsymbol{x}\right\|^2}{k}.$$

Let (Q,u) be the output of a call to Monotonic-Coreset(P,ε,δ,k)*; see Algorithm 1.*

Then, with probability at least $1-\delta$, (Q,u) is an ε-coreset for $(P,\mathbf{1},\mathbb{R}^d,c_{\text{sigmoid},k})$, i.e., for every $x\in\mathbb{R}^d$

$$\left|\sum_{p\in P} c_{\text{sigmoid},k}(\boldsymbol{p},\boldsymbol{x}) - \sum_{p\in Q} u(p)c_{\text{sigmoid},k}(\boldsymbol{p},\boldsymbol{x})\right| \leq \varepsilon\sum_{p\in P} c_{\text{sigmoid},k}(\boldsymbol{p},\boldsymbol{x}).$$

Moreover, for $t=(1+\sqrt{k})\log n$,

$$|Q| \in O\left(\frac{t}{\varepsilon^2}\left(d\log t + \log\frac{1}{\delta}\right)\right)$$

and (Q,u) can be computed in $O(dn+n\log n)$ time.

Proof. By [18], the dimension of $(P,w,\mathbb{R}^d,c)$ is at most $d+1$, where (P,w) is a weighted set, $P\subseteq\mathbb{R}^d$, and $c(p,x)=f(\boldsymbol{p}\cdot\boldsymbol{x})$ for some monotonic and invertible function f. By Lemma 12, the total sensitivity of $(P,\mathbf{1},\mathbb{R}^d,c_{\text{sigmoid},k})$ is bounded by

$$t \in O\left(\log n + \sqrt{k}\sum_{j=1}^{n}\frac{\left\|p_j\right\|}{j}\right) = O\left(\log n + \sqrt{k}\sum_{j=1}^{n}\frac{1}{j}\right) = O\left((1+\sqrt{k})\log n\right),$$

where the last equality holds since the input points are in the unit ball.

Plugging these upper bounds on the dimension and total sensitivity of the query space in Theorem 5, yields that a call to Algorithm 1, which samples points from P based on their sensitivity bound, returns the desired coreset (Q,u). The running time is dominated by sorting the length of the points in $O(n\log n)$ time after computing them in $O(nd)$ time. □

10 Example: Coreset for Logistic Regression

We show that our framework can be used for construction a coreset for logistic regression.

Lemma 14. *Let $f = \log(1 + e^x)$ for every $x \in \mathbb{R}$ and let $c > 0$. Then, there is $k_0 > 0$ such that for every $k \geq k_0$ and for every $0 \leq x \leq R$*

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 3\frac{\log\left(2e^{cR}\right)}{\log(2)}\sqrt{k}c.$$

Proof. See Lemma 22 in the appendix. □

Lemma 15. *Let $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subset \mathbb{R}^d$ be a set of points, sorted by their length. I.e. $\|\boldsymbol{p}_i\| \leq \|\boldsymbol{p}_j\|$ for every $1 \leq i \leq j \leq n$. Let $R, k > 0$ and $c_{\text{logistic},k}(\boldsymbol{p}, \boldsymbol{x}) = \log(1 + e^x) + \frac{\|\boldsymbol{x}\|^2}{k}$ for every $\boldsymbol{x} \in B(\mathbf{0}, R)$ and $\boldsymbol{p} \in P$. Denote by $B(\mathbf{0}, R)$ the ball of radius R centered at the origin. Then the sensitivity of every $p_j \in P$ is bounded by $s(\boldsymbol{p}) = s_{P,\mathbf{1},B(\mathbf{0},R),c_{\text{logistic},k}}(\boldsymbol{p}) \in O\left(\frac{R^2\|\boldsymbol{p}_j\|\sqrt{k}+R}{j}\right)$, and the total sensitivity is*

$$t = \sum_{p \in P} s(p) \in O\left(R\log n + R^2\sqrt{k}\sum_{j=1}^{n}\frac{\|p_j\|}{j}\right).$$

Proof. Define $f(z) = \log(1 + e^z)$ and $g(z) = z^2$ for every $z \in \mathbb{R}$. Let $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{p}_j \in P$ and $i \in [1, j]$ be an integer. We substitute $c = \|\boldsymbol{p}_i\|$ in Lemma 22 to obtain that for every $z > 0$

$$\frac{f(\|\boldsymbol{p}_i\| x) + \frac{x^2}{k}}{f(-\|\boldsymbol{p}_i\| x) + \frac{x^2}{k}} \leq 3\frac{\log\left(2e^{\|\boldsymbol{p}_i\|R}\right)}{\log(2)}\sqrt{k}\|\boldsymbol{p}_i\|.$$

Denote $b_{\boldsymbol{p}_i} = 3\frac{\log\left(2e^{\|\boldsymbol{p}_i\|R}\right)}{\log(2)}\sqrt{k}\|\boldsymbol{p}_i\|$ and multiply the above term by $f(-\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}$ to get

$$f(\|\boldsymbol{p}_i\| z) + \frac{z^2}{k} \leq b_{\boldsymbol{p}_i}\left(f(-\|\boldsymbol{p}_i\| z) + \frac{z^2}{k}\right).$$

Substituting in Lemma 9 $\boldsymbol{p} = \boldsymbol{p}_i, f(z) = \log(1 + e^z), g(z) = z^2, M = \log(1 + e^R), f(0) = \log(2)$ yields

$$\begin{aligned}&\max_{\boldsymbol{p}' \in P} c_{\text{logistic},k}(\boldsymbol{p}', \boldsymbol{x}) \leq \\ &\frac{\log(1 + e^R)(b_{\boldsymbol{p}_i} + 1)c_{\text{logistic},k}(\boldsymbol{p}_i, \boldsymbol{x})}{log(2)}.\end{aligned} \tag{25}$$

Thus

$$c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x}) \leq \max_{\boldsymbol{p}' \in P} c_{\text{logistic},k}(\boldsymbol{p}', \boldsymbol{x}) \leq \tag{26}$$

$$\frac{\log(1 + e^R)(b_{\boldsymbol{p}_i} + 1)c_{\text{logistic},k}(\boldsymbol{p}_i, \boldsymbol{x})}{log(2)}, \tag{27}$$

where (26) is since $\boldsymbol{p}_j \in P$ and (27) is by (25). Dividing both sides by $\frac{\log(1+e^R)}{log(2)}(b_{\boldsymbol{p}_i} + 1)$ yields

$$c_{\text{logistic},k}(\boldsymbol{p}_i, \boldsymbol{x}) \geq \frac{c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x})}{\frac{\log(1+e^R)}{log(2)}(b_{\boldsymbol{p}_i} + 1)}. \tag{28}$$

We now proceed to bound the sensitivity of $\boldsymbol{p}_j$. Since the set of points $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_j\}$ is a subset of P, and since the cost function $c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x})$ is positive we have that

$$\sum_{\boldsymbol{p'} \in P} c_{\text{logistic},k}(\boldsymbol{p'}, \boldsymbol{x}) \geq \sum_{i=1}^{j} c_{\text{logistic},k}(\boldsymbol{p}_i, \boldsymbol{x}). \tag{29}$$

By summing (28) over $i \leq j$, we obtain

$$\begin{aligned} &\sum_{i=1}^{j} c_{\text{logistic},k}\boldsymbol{p}_i, \boldsymbol{x}) \geq \\ &c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x}) \sum_{i=1}^{j} \frac{\log(2)}{\log(1+e^R)(b_{\boldsymbol{p}_i}+1)} \geq \\ &c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x}) \frac{j\log(2)}{\log(1+e^R)(b_{\boldsymbol{p}_j}+1)}, \end{aligned} \tag{30}$$

where the last inequality holds since $b_{p_i} = 3\frac{\log(1+e^R)}{log(2)} \left\|p_i\right\| \sqrt{k} \leq b_{\boldsymbol{p}_j}$ for every $i \leq j$. Combining (29) and (30) yields

$$\sum_{\boldsymbol{p'} \in P} c_{\text{logistic},k}(\boldsymbol{p'}, \boldsymbol{x}) \geq \frac{j\log(2) c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x})}{\log(1+e^R)(b_{\boldsymbol{p}_j}+1)} \tag{31}$$

Therefore, the sensitivity is bounded by

$$\begin{aligned} &s_{P,\mathbf{1},B(\mathbf{0},R),c_{\text{logistic},k}}(\boldsymbol{p}_j) = \\ &\sup_{\boldsymbol{x} \in B(\mathbf{0},R)} \frac{c_{\text{logistic},k}(\boldsymbol{p}_j, \boldsymbol{x})}{\sum_{\boldsymbol{p'} \in P} c_{\text{logistic},k}(\boldsymbol{p'}, \boldsymbol{x})} \leq \\ &\frac{\log(1+e^R)(b_{\boldsymbol{p}_j}+1)}{j\log(2)} \leq \\ &\frac{\log(1+e^R)\left(\frac{3\log(1+e^R)}{log(2)} \left\|p_j\right\| \sqrt{k}+1\right)}{j\log(2)}. \end{aligned}$$

Thus, $s_{P,\mathbf{1},B(\mathbf{0},R),c_{\text{logistic},k}}(\boldsymbol{p}_j) \in O\left(\frac{R^2\left\|\boldsymbol{p}_j\right\|\sqrt{k}+R}{j}\right)$. Summing this sensitivity bounds the total sensitivity by

$$\sum_{j=1}^{n} \frac{R^2\left\|\boldsymbol{p}_j\right\|\sqrt{k}+R}{j} \in O\left(R\log n + R^2\sqrt{k}\sum_{j=1}^{n}\frac{\left\|p_j\right\|}{j}\right).$$

□

Theorem 16. *Let P be a set of n points in the unit ball of $\mathbb{R}^d$, $\varepsilon, \delta \in (0,1)$, and $R, k > 0$. For every $\boldsymbol{p} \in \mathbb{R}^d$,$\boldsymbol{x} \in B(\mathbf{0}, R)$ let*

$$c_{\text{logistic},k}(\boldsymbol{p}, \boldsymbol{x}) = \log\left(1 + e^{\boldsymbol{p}\cdot\boldsymbol{x}}\right) + \frac{\left\|\boldsymbol{x}\right\|^2}{k}.$$

Let (Q, u) be the output of a call to Monotonic-Coreset$(P, \varepsilon, \delta, k)$.

Then, with probability at least $1-\delta$, (Q, u) is an ε-coreset for $(P, \mathbf{1}, \mathbb{R}^d, c_{\text{logistic},k})$, i.e., for every $x \in \mathbb{R}^d$

$$\begin{aligned} &\left|\sum_{p \in P} c_{\text{logistic},k}\left(\boldsymbol{p}, \boldsymbol{x}\right) - \sum_{p \in Q} u(p) c_{\text{logistic},k}\left(\boldsymbol{p}, \boldsymbol{x}\right)\right| \\ &\leq \varepsilon \sum_{p \in P} c_{\text{logistic},k}\left(\boldsymbol{p}, \boldsymbol{x}\right). \end{aligned}$$

Moreover, for $t = R \log n(1 + R\sqrt{k})$,

$$|Q| \in O\left(\frac{t}{\varepsilon^2}\left(d \log t + \log \frac{1}{\delta}\right)\right)$$

and (Q, u) *can be computed in* $O(dn + n \log n)$ *time.*

Proof. By [18], the dimension of $(P, w, \mathbb{R}^d, c)$ is at most $d + 1$, where (P, w) is a weighted set, $P \subseteq \mathbb{R}^d$, and $c(p, x) = f(\boldsymbol{p} \cdot \boldsymbol{x})$ for some monotonic and invertible function f. By Lemma 15, the total sensitivity of $(P, \mathbf{1}, \mathbb{R}^d, c_{\text{logistic},k})$ is bounded by

$$t \in O\left(R \log n + R^2\sqrt{k} \sum_{j=1}^{n} \frac{\|p_j\|}{j}\right) =$$

$$O\left(R \log n + R^2\sqrt{k} \sum_{j=1}^{n} \frac{1}{j}\right) = O\left(R \log n(1 + R\sqrt{k})\right),$$

where the last equality holds since the input points are in the unit ball.

Plugging these upper bounds on the dimension and total sensitivity of the query space in Theorem 5, yields that a call to MONOTONIC-CORESET, which samples points from P based on their sensitivity bound, returns the desired coreset (Q, u). The running time is dominated by sorting the length of the points in $O(n \log n)$ time after computing them in $O(nd)$ time. Sampling $m = |Q|$ points from n points according to such a given distribution takes $O(1)$ time after pre-processing of $O(n)$ time. □

11 Experimental Results

We implemented Algorithm 1 and run it on both synthetic and real-world datasets as explained below. To be consistent with the theoretical results, we apply the algorithm specifically on the sigmoid function, and as common in related coresets papers, we also focus on the traditional off-line construction. This is since the approximation error of the streaming, distributed and dynamic versions of such coreset constructions is based on running the off-line versions independently on multiple subsets of the data, merge the resulting coreset and reduce them again recursively. We leave these extensions to the full version of the paper.

Open code. For the benefit of the community, and for reproducing our experimental results, our code is open under the GPL license and all the experiments are reproducible.

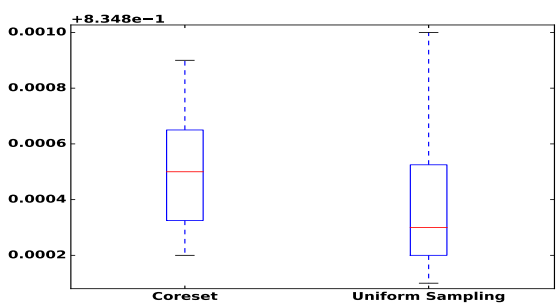
11.1 Minimizing Sum of Sigmoids

We used 2 datasets from the UCI repository Lichman (2013) and a synthetic data for our experiments.

Synthetic dataset. This data contains a set of $n = 20,010$ points in $\mathbb{R}^2$. $20,000$ of the points were generated by sampling a two dimensional normal distribution with mean $\mu_1 = (10,000, 10,000)$ and covariance matrix $\Sigma_1 = \left(\begin{smallmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{smallmatrix}\right)$ and 10 points were generated by sampling a two dimensional normal distribution with mean $\mu_2 = (-9998, -9998)$ and covariance matrix $\Sigma_2 = \left(\begin{smallmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{smallmatrix}\right)$

Bank marketing dataset [22] [1] consists $n = 20,000$ records. Each record is a $d = 10$ dimensional vector with numerical values. The data was generated for direct marketing campaigns of a Portuguese banking institution. Each record represents a marketing call to a client, that aims to convince him/her to buy a product (bank term deposit). A binary label (yes or no) was added to each record. We used the numerical

[1] `https://archive.ics.uci.edu/ml/datasets/bank+marketing`



(a) Error box-plots for 35,000 points sampled from the Cifar10 dataset using uniform sampling and our coreset sampling scheme.

(b) The success rate for classifying the Cifar10 dataset.

Figure 1: Using Coresets to boost Cifar10 classification

values of the records to predict if a subscription was made.This dataset was also used for experimentation in [29, 4]. The challenge is to compute a classification model for this supervised data, that gets a record that represents a potential client as an input, and returns the binary label as an output, or more generally, an estimated probability of the event that the client would buy the product. Such a model may also tell the company the connection between the features of the client and the outcome of the call, as well as the importance of each feature in the decision.

Wine Quality dataset [5] [2] This dataset contains records of physicochemical and sensory data about the red and white variants of the Portuguese "Vinho Verde" win. Each record in the dataset is a $d = 12$ dimensional numerical feature vector. Each record in the dataset is labeled 'white' or 'red'. The total number of samples is $n = 6497$. This dataset was also used for experimentation in [28, 7, 19]. The data is the results of a chemical analysis of wines grown in the same region in Italy but derived from different cultivars. The analysis determined the quantities of 13 constituents found in each of the types of wines. The goal is to train a classification model for this labeled data, which gets the chemical data of a wine sample as an input ad returns the binary label as an output.

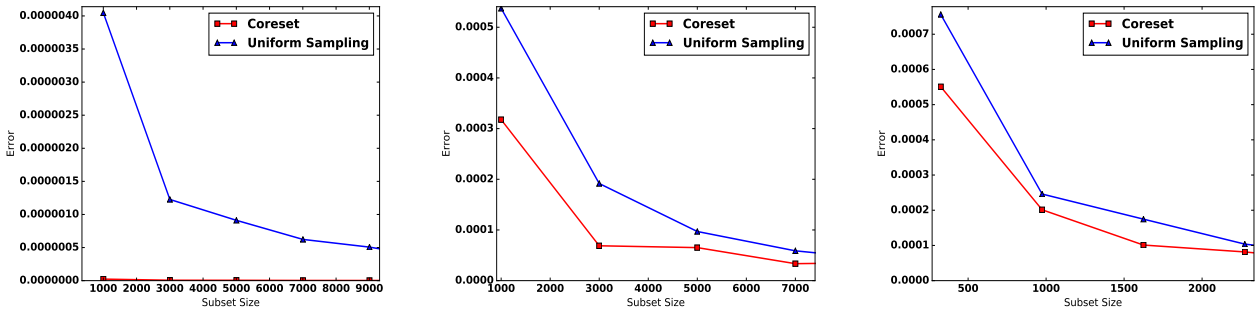
11.1.1 Experimental Setup

For a given size m we computed a coreset of size m using Algorithm 1. We used the datasets above to produce coresets of size $0.1n \leq m \leq 0.9n$, where n is the size of the full data, then we normalized the data and found the optimal solution to the problem with values of $k = 100, 500, 1000$ using the BFGS algorithm. We repeated the experiment with a uniform sample of size m. For each optimal solution that we have found, we computed the sum of sigmoids and denoted these "approximated solutions" by C_1 and C_2 for our algorithm and uniform sampling respectively. The "ground truth" C^k was computed using BFGS on the entire dataset. The empirical error is then defined to be $E_t = \left|\frac{C_t}{C^k} - 1\right|$ for $t = 1, 2$. For every size m we computed E_1 and E_2 100 times and calculated the mean of the results.

11.2 Convolutional Neural Networks

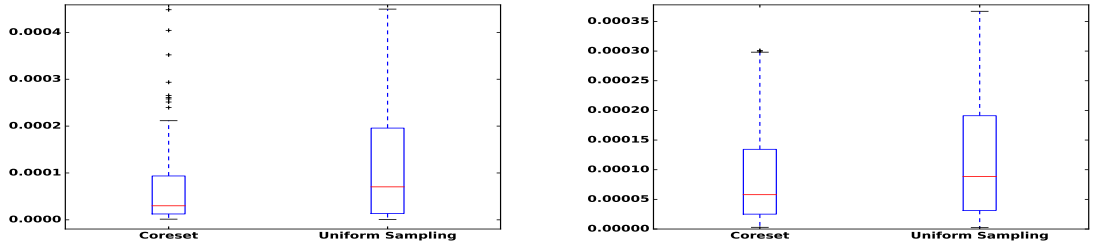
While we do not have coreset for the set of complete neural networks, we suggest a novel technique to use coresets for improving existing state-of-the art networks. In this network, the layers except from the last one get the original input and produce a set P of the input with different features that correspond to the neurons in this layer. By computing coreset Q for P and the sigmoid function, we can train each neuron much faster by running existing heuristic for sigmoid optimization on Q. The result is a suggestion for a

[2] `https://archive.ics.uci.edu/ml/datasets/wine+quality`



(a) Synthetic dataset, $k = 100$ (b) Bank Marketing dataset, $k = 500$ (c) Wine dataset, $k = 1000$

Figure 2: Comparison between uniform sampling and our coreset.



(a) Bank Marketing dataset, $k = 500$ (b) Wine dataset, $k = 1000$

Figure 3: Error box-plots for $25,000$ points sampled from the real datasets using uniform sampling and our coreset sampling scheme.

new weights for the last layer of the network. For our experiments we used the Cifar10 dataset[20], which consists of $60,000$ 32×32 color images in 10 classes with 6000 images per class. We used $50,000$ points as training data and $10,000$ points for testing data. For $m = 5,000, 15,000, 25,000, 35,000$, we sampled m points from P using our implementation of Algorithm 1, and trained the a new softmax output layer using the sampled subset. We then used the testing data to test the performance of the network. We compared our algorithm to uniform sampling and to the success rate of the original set. We repeated every experiment 10 times and calculated the average of the results.

11.3 Results

Figure 2 depicts results for the sigmoid experiment It can be seen that our sampling algorithm outperforms uniform sampling. Important to note, that our algorithm starts with small error value compared to others and improves error value gradually with sample size, while two others starts with greater error values and succeeds to converge to smaller values only for large sample subsets

Figure 3 shows the box-plot of error distribution for the 100 experiments performed with $25,000$ points subsets of the Wine and Bank marketing datasets. It can be seen that the variance of our algorithm is considerably smaller then the variance of the uniform sampling scheme.

The results of the Cifar10 classification are depicted in Figure 1(b). It can be seen that out coreset outperform the results obtained by uniform sampling and the results obtained by training the network on the original dataset. These results suggest that much can be gained from using coresets for training of NN.

Figure 1(a) shows the box-plot of error distribution for the 10 experiments performed with $35,000$ points subsets of the Cifar10 dataset.

References

[1] Coreset for monotonic functions with applications to deep learning full version, 2018.

[2] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal coresets for least-squares regression. *IEEE transactions on information theory*, 59(10):6880–6892, 2013.

[3] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

[4] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pages 5036–5044, 2017.

[5] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[6] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for \ell_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

[7] Gal Elidan. Copula bayesian networks. In *Advances in neural information processing systems*, pages 559–567, 2010.

[8] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in neural information processing systems*, pages 2142–2150, 2011.

[9] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.

[10] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18. ACM, 2007.

[11] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. SIAM, 2013.

[12] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013.

[13] Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. Society for Industrial and Applied Mathematics, 2012.

[14] Dan Feldman and Tamir Tassa. More constraints, smaller coresets: constrained matrix approximation of sparse big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*, pages 249–258. ACM, 2015.

[15] Sariel Har-Peled. Coresets for discrete integration and clustering. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 33–44. Springer, 2006.

[16] Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In *IJCAI*, pages 836–841, 2007.

[17] J. Hellerstein. Parallel programming in the age of big data. Gigaom Blog. 9th November, 2008.

[18] Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *Advances In Neural Information Processing Systems*, pages 4080–4088, 2016.

[19] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. A convex formulation for learning from crowds. *Transactions of the Japanese Society for Artificial Intelligence*, 27(3):133–142, 2012.

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[21] M. Langberg and L. J. Schulman. Universal ε approximators for integrals. *To appear in proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

[22] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[23] T. Segaran and J. Hammerbacher. *Beautiful Data: The Stories Behind Elegant Data Solutions*. O'Reilly Media, 2009.

[24] Jiří Šíma. Training a single sigmoidal neuron is hard. *Training*, 14(11), 2006.

[25] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

[26] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl.*, 16:264–280, 1971.

[27] Kasturi Varadarajan and Xin Xiao. A near-linear algorithm for projective clustering integer points. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1329–1342. SIAM, 2012.

[28] Shusen Wang and Zhihua Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[29] Peifeng Yin, Ping Luo, and Taiga Nakamura. Small batch or large batch: Gaussian walk with rebound can teach. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2017.

[30] Yan Zheng and Jeff M Phillips. Coresets for kernel regression. *arXiv preprint arXiv:1702.03644*, 2017.

12 Appendix

Lemma 17. *Let $f:\mathbb{R}\to(0,\infty)$ be a monotonic increasing function such that $f(0)>0$. Let $c,k>0$. There is exactly one number $x_{kc}>0$ that simultaneously satisfies the following claims.*

(i) $f\left(-c\sqrt{k}x_{kc}\right)=x_{kc}^2$.

(ii) *For every $x>0$, if $f\left(-c\sqrt{k}x\right)>x^2$ then $x<x_{kc}$.*

(iii) *For every $x>0$, if $f\left(-c\sqrt{k}x\right)<x^2$ then $x>x_{kc}$.*

(iv) *There is $k_0>0$ such that for every $k\geq k_0$*

$$\frac{1}{x_{kc}}\leq c\sqrt{k}.$$

Proof. Let $g(x)=x^2$. Define

$$h_{kc}(x)=f(-c\sqrt{k}x)-g(x). \tag{32}$$

(i): It holds that

$$h_{kc}(0)=f(0) \tag{33}$$

and

$$h_{kc}\left(\sqrt{f(0)+1}\right)<0, \tag{34}$$

where (34) holds since $f\left(-c\sqrt{k}x\right)\leq f(0)$ for every $x>0$, and $g\left(\sqrt{f(0)+1}\right)=f(0)+1$. From (33) and (34) we have that $0\in\left[h_{kc}\left(\sqrt{f\left(0\right)+1}\right),h_{kc}(0)\right]$. Using the Intermediate Value Theorem (Theorem 23) we have that there is $x_1\in\left(0,\sqrt{f(0)+1}\right)$ such that

$$h_{kc}(x_1)=0. \tag{35}$$

We prove that x_1 is unique. By contradiction. Assume that there is $x_2\neq x_1$ such that

$$h_{kc}(x_1)=h_{kc}(x_2)=0. \tag{36}$$

Wlog assume that $x_1<x_2$. By The Mean Value Theorem (Theorem 24), there is $r\in(x_1,x_2)$ such that

$$h'_{kc}(r)=\frac{h_{kc}(x_2)-h_{kc}(x_1)}{x_2-x_1} \tag{37}$$

$$=0, \tag{38}$$

where (38) is by (36). The derivative of h_{kc} is

$$h'_{kc}(x)=\left(f\left(-c\sqrt{k}x\right)-g(x)\right)' \tag{39}$$

$$=-c\sqrt{k}f'\left(-c\sqrt{k}x\right)-g'(x)<0, \tag{40}$$

where (39) is by (32) and (40) is since f is monotonic increasing and thus $f'(x)>0$ for every $x\in\mathbb{R}$ and $x,k,c>0$. (40) is a contradiction to (38). Thus the Assumption (36) is false and x_1 is unique.

By (32) and (35)

$$f\left(-c\sqrt{k}x_1\right)=g(x_1). \tag{41}$$

By letting $x_{kc} = x_1$ and recalling that $g(x) = x^2$ we obtain

$$f\left(-c\sqrt{k}x_{kc}\right) = x_{kc}^2.$$

(ii): Let $x > 0$ such that $f\left(-c\sqrt{k}x\right) > x^2$. Plugging this and the definition $g(x) = x^2$ in (32) yields

$$h_{kc}(x) > 0. \tag{42}$$

We already proved that $h'_{kc}(x) < 0$ always. By the Inverse of Strictly Monotone Function Theorem (Theorem 25) we have that the inverse h_{kc}^{-1} of h_{kc} is a strictly monotone decreasing function. Applying h_{kc}^{-1} on both sides of (42) gives

$$x < x_{kc}.$$

(iii): Let $x > 0$ such that $f\left(-c\sqrt{k}x\right) < x^2$. By this and by the definition of g and (32) we have

$$h_{kc}(x) < 0. \tag{43}$$

We already proved that $h'_{kc}(x) < 0$ always. By the Inverse of Strictly Monotone Function Theorem (Theorem 25) we have that h_{kc} has a strictly monotone decreasing inverse function h_{kc}^{-1}. Applying h_{kc}^{-1} on both sides of (43) gives

$$x > x_{kc}.$$

(iv): We need to prove that there is k_0 such that for every $k > k_0$ we have

$$x_{kc} \geq \frac{1}{c\sqrt{k}} \tag{44}$$

By contradiction, assume that

$$x_{kc} < \frac{1}{c\sqrt{k}}. \tag{45}$$

It holds that

$$f\left(-c\sqrt{k}x_{kc}\right) > f(-1). \tag{46}$$

where (46) holds since f is increasing and by (45) $-c\sqrt{k}x_{kc} > -1$. Since $\lim_{k\to\infty} \frac{1}{c^2 k} = 0$, there is $k_0 > 0$ such that for every $k > k_0$

$$\begin{aligned} f(-1) &> \frac{1}{c^2 k} \\ &> x_{kc}^2, \end{aligned} \tag{47}$$

where (47) is by (45). Plugging (47) in (46) yields

$$f\left(-c\sqrt{k}x_{kc}\right) > x_{kc}^2. \tag{48}$$

In contradictions to **(i)**. Thus

$$x_{kc} \geq \frac{1}{k\sqrt{c}} \tag{49}$$

□

Lemma 18. *Let f be as in Lemma 17 and let $x_{1,1} > 0$ which is obtained by applying Lemma 17**(i)** with f and $k = c = 1$. Then, For every $x \geq 0$*

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}.$$

Proof. Let $x \geq 0$. Substituting $k = c = 1$ in Lemma 17**(i)** yields that $f(-x_{1,1}) = x_{1,1}^2$. We show that $\frac{f(x)+x^2}{f(-x)+x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}$ via the following case analysis. **(i)** $f(x) \geq x^2$, **(ii)** $f(x) \geq x^2$, **(iii)** $f(x) < x^2$and $f(-x) \geq x^2$, and **(iv)** $f(x) < x^2$and $f(-x) < x^2$.

Case (i): $f(x) \geq x^2$ and $f(-x) \geq x^2$. Since $f(-x) \geq x^2$, by substituting $k = c = 1$ in Lemma 17**(ii)**, we have that $x \leq x_{1,1}$. Hence

$$\begin{aligned} f(-x) + x^2 \geq & f(-x) && (50)\\ \geq & f(-x_{1,1}) && (51)\\ = & x_1^2, && (52) \end{aligned}$$

where (50) is since $x^2 > 0$, (51) is since f is increasing and $x \leq x_{1,1}$, and (52) is by definition of $x_{1,1}$. By adding $f(x)$ to both sides of the assumption $f(x) \geq x^2$ of Case (i) we obtain

$$2f(x) \geq f(x) + x^2. \tag{53}$$

By (53) and (52) we obtain

$$\frac{f(x)+x^2}{f(-x)+x^2} \leq \frac{2f(x)}{x_{1,1}^2} \leq \frac{2}{x_{1,1}^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}. \tag{54}$$

Case (ii): $f(x) \geq x^2$ and $f(-x) < x^2$. Since $f(-x) < x^2$, substituting $k = c = 1$ in Lemma 17**(iii)**, there is $x_{1,1}$ such that

$$\begin{aligned} f(-x) + x^2 \geq & x^2 && (55)\\ > & x_{1,1}^2, && (56) \end{aligned}$$

where (55) is since f is a positive function and (56) is since $x > x_{1,1}$. By adding $f(x)$ to both sides of the assumption $f(x) \geq x^2$ of Case (ii) we have that

$$f(x) + x^2 \leq 2f(x). \tag{57}$$

By (57)) and (56) we obtain

$$\frac{f(x)+x^2}{f(-x)+x^2} \leq \frac{2f(x)}{x_{1,1}^2} \leq \frac{2}{x_{1,1}^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}. \tag{58}$$

Case (iii): $f(x) < x^2$and $f(-x) \geq x^2$. By adding x^2 to both sides of the assumption $f(x) < x^2$ of Case (iii) we have that

$$f(x) + x^2 \leq 2x^2. \tag{59}$$

Furthermore, since $f(-x) > 0$ we have that

$$f(-x) + x^2 \geq x^2. \tag{60}$$

Combining (59) and (60) we obtain

$$\frac{f(x)+x^2}{f(-x)+x^2} \leq \frac{2x^2}{x^2} \leq 2 \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}. \tag{61}$$

Case (iv): $f(x) < x^2$ and $f(-x) < x^2$. By adding x^2 to both sides of the assumption $f(x) < x^2$ of Case (iv) we have that

$$f(x) + x^2 \leq 2x^2. \tag{62}$$

Furthermore, since $f(-x) > 0$ we have that

$$f(-x) + x^2 \geq x^2. \tag{63}$$

Combining (62) and (63) we obtain

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \frac{2x^2}{x^2} \leq 2 \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}. \tag{64}$$

Combining the results of the case analysis: (54), (58), (61),and (64) we have that

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}. \tag{65}$$

□

Lemma 19. *Let f be as in Lemma 17, $x_{1,1}$ as in Lemma 18 and $c > 0$. Assume that there is $D > 1$ such that $\frac{f(cy)}{f\left(\frac{y}{\sqrt{k}}\right)} < D$ for every $y \geq 0$. Then, there is $k_0 > 0$ such that for every $k \geq k_0$ and for every $x \geq 0$,*

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 3D \max\left\{2, \frac{2}{x_{1,1}^2}\right\} c\sqrt{k}.$$

Proof. Let $x \geq 0$ and $k, c > 0$. We have that

$$f(cx) + \frac{x^2}{k} \leq Df\left(\frac{x}{\sqrt{k}}\right) + \frac{x^2}{k} \tag{66}$$

$$\leq D \max\left\{2, \frac{2}{x_{1,1}^2}\right\}\left(f\left(-\frac{x}{\sqrt{k}}\right) + \frac{x^2}{k}\right), \tag{67}$$

where (66) holds since $\frac{f(cy)}{f\left(\frac{y}{\sqrt{k}}\right)} < D$ for every $y \geq 0$ and (67) holds since $\frac{x^2}{k} \leq D\frac{x^2}{k}$, and since, by Lemma 18, for every positive z we have that

$$\frac{f(z) + z^2}{f(-z) + z^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\}.$$

Dividing (67) by $f(-cx) + \frac{x^2}{k}$ yields

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq D \max\left\{2, \frac{2}{x_{1,1}^2}\right\}\left(\frac{f\left(-\frac{x}{\sqrt{k}}\right) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}}\right). \tag{68}$$

We now proceed to bound $R_{ck} = \frac{f\left(-\frac{x}{\sqrt{k}}\right) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}}$. By denoting $z = \frac{x}{\sqrt{k}}$ we have that

$$R_{ck} = \frac{f(-z) + z^2}{f\left(-c\sqrt{k}z\right) + z^2}. \tag{69}$$

We now compute an upper bound for R_{ck} using the following case analysis: **(i)** $f(-z) \geq z^2$ and $f\left(-c\sqrt{k}z\right) \geq z^2$, **(ii)** $f(-z) < z^2$ and $f\left(-c\sqrt{k}z\right) < z^2$, **(iii)**, and **(iv)** $f(-z) < z^2$ and $f\left(-c\sqrt{k}z\right) \geq z^2$. Let $z_{ck} > 0$ be such that $f\left(-c\sqrt{k}z_{ck}\right) = z_{ck}^2$ as given by Lemma 17**(i)**. There are four cases

Case (i): $f(-z) \geq z^2$ and $f\left(-c\sqrt{k}z\right) \geq z^2$. Since $f\left(-c\sqrt{k}z\right) \geq z^2$, by Lemma 17**(iii)** we have that $z \leq z_{ck}$. Thus

$$f\left(-c\sqrt{k}z\right) \geq f\left(-c\sqrt{k}z_{ck}\right) \tag{70}$$

$$= z_{ck}^2, \tag{71}$$

where (70) holds since f is monotonic and $z \leq z_{ck}$, and (71) is from the definition of z_{ck}. Furthermore, by adding $f(-z)$ to both sides of the assumption $f(-z) \geq z^2$, we have that

$$f(-z) + z^2 \leq 2f(-z). \tag{72}$$

Substituting (72) and (71) in (69) yields

$$R_{ck} = \frac{f(-z) + z^2}{f\left(-c\sqrt{k}z\right) + z^2} \leq \frac{2f(-z)}{z_{ck}^2} \leq \frac{1}{z_{ck}^2}, \tag{73}$$

where the last inequality, is since $f(-z) \leq 1/2$ for every $z \geq 0$.

Case (ii): $f(-z) < z^2$ and $f\left(-c\sqrt{k}z\right) < z^2$. By adding z^2 to both sides of the assumption $f(-z) < z^2$, we have that

$$f(-z) + z^2 \leq 2z^2. \tag{74}$$

Furthermore, since $f\left(-c\sqrt{k}z\right) > 0$ we have that

$$f\left(-c\sqrt{k}z\right) + z^2 \geq z^2. \tag{75}$$

Combining (74) and (75) yields

$$R_{ck} = \frac{f(-z) + z^2}{f\left(-c\sqrt{k}z\right) + z^2} \leq \frac{2z^2}{z^2} = 2. \tag{76}$$

Case (iii): $f(-z) \geq z^2$ and $f\left(-c\sqrt{k}z\right) < z^2$. Since $f\left(-c\sqrt{k}z\right) < z^2$, by Lemma 17 we have that $z > z_{ck}$. Thus

$$f\left(-c\sqrt{k}z\right) + z^2 \geq z^2 \geq z_{ck}^2. \tag{77}$$

By adding $f(-z)$ to both sides of the assumption $f(-z) \geq z^2$, we have that

$$2f(-z) \geq f(-z) + z^2. \tag{78}$$

Substituting (77) and (78) in (69) yields

$$R_{ck} = \frac{f(-z) + z^2}{f\left(-c\sqrt{k}z\right) + z^2} \leq \frac{2f(-z)}{z_{ck}^2} \leq \frac{1}{z_{ck}^2}. \tag{79}$$

Case (iv): $f(-z) < z^2$ and $f\left(-c\sqrt{k}z\right) \geq z^2$. By adding z^2 to both sides of the assumption $f(-z) < z^2$, we have that

$$f(-z) + z^2 \leq 2z^2. \tag{80}$$

Since $f\left(-c\sqrt{k}z\right) > 0$ we have that

$$f\left(-c\sqrt{k}z\right) + z^2 > z^2. \tag{81}$$

Plugging (80) and (81) in (69) yields

$$R_{ck} = \frac{f(-z) + z^2}{f\left(-c\sqrt{k}z\right) + z^2} \leq \frac{2z^2}{z^2} = 2. \tag{82}$$

Combining the results of the case analysis: (73), (76), 79,and (82) we have that

$$R_{ck} \leq 2 + \frac{1}{z_{ck}^2}. \tag{83}$$

By Lemma 17**(iv)** we have that there is $k_0 > 0$ such that for every $k \geq k_0$,

$$\frac{1}{z_{ck}^2} \leq c\sqrt{k}. \tag{84}$$

Substituting (84) in (83) yields

$$R_{ck} \leq 2 + c\sqrt{k}, \tag{85}$$

by (68) we have

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq D \max\left\{2, \frac{2}{x_{1,1}^2}\right\} R_{ck}.$$

Substituting (85) in the last term gives

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq D \max\left\{2, \frac{2}{x_{1,1}^2}\right\} \left(2 + c\sqrt{k}\right).$$

It holds that for every $k \geq \frac{1}{c^2}$ we have $2 \leq 2c\sqrt{k}$ plugging this in the above term yields

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 3D \max\left\{2, \frac{2}{x_{1,1}^2}\right\} c\sqrt{k}.$$

□

Lemma 20. *Let $f = \frac{1}{1+e^{-x}}$ for every $x \in \mathbb{R}$ and let $c > 0$. Then, there is $k_0 > 0$ such that for every $k \geq k_0$ and for every $x \geq 0$*

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 66c\sqrt{k}$$

Proof. It holds that $f(0) > 0$. Applying Lemma 17 with $k = c = 1$ yields $x_{1,1}$ such that $f(-x_{1,1}) = x_{1,1}^2$. We now bound $x_{1,1}$. Calculation shows that

$$f\left(-\sqrt{\ln(1.2)}\right) > \left(\sqrt{\ln(1.2)}\right)^2.$$

Plugging $x = \sqrt{\ln(1.2)}, k = 1, c = 1$ in Lemma 17**(ii)** yields

$$x_1 \geq \sqrt{\ln(1.2)}. \tag{86}$$

By applying Lemma 18 with f we have

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\} \leq 11, \tag{87}$$

where the last inequality is by (94).

For every $c, k > 0$ it holds that

$$\frac{f(cx)}{f\left(\frac{x}{\sqrt{k}}\right)} \leq 2, \tag{88}$$

where (92) holds since for every $y > 0$ $f(y) \leq 1$ and $f\left(\frac{x}{\sqrt{k}}\right) \geq \frac{1}{2}$. Applying Lemma 19 with $f, D = 2$ yields

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 66\sqrt{k}c. \tag{89}$$

□

Lemma 21. *Let $f = \left(\frac{1}{1+e^{-x}}\right)^2$ for every $x \in \mathbb{R}$ and let $c > 0$. Then, there is $k_0 > 0$ such that for every $k \geq k_0$ and for every $x \geq 0$*

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 168c\sqrt{k}$$

Proof. It holds that $f(0) > 0$. Applying Lemma 17 with $k = c = 1$ yields $x_{1,1}$ such that $f(-x_{1,1}) = x_{1,1}^2$. We now bound $x_{1,1}$. Calculation shows that

$$f\left(-\sqrt{\ln(1.15)}\right) > \left(\sqrt{\ln(1.15)}\right)^2.$$

Plugging $x = \sqrt{\ln(1.15)}, k = 1, c = 1$ in Lemma 17**(ii)** yields

$$x_1 \geq \sqrt{\ln(1.15)}. \tag{90}$$

By applying Lemma 18 with f we have

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\} \leq 14, \tag{91}$$

where the last inequality is by (94).

For every $c, k > 0$ it holds that

$$\frac{f(cx)}{f\left(\frac{x}{\sqrt{k}}\right)} \leq 4, \tag{92}$$

where (92) holds since for every $y > 0$ $f(y) \leq 1$ and $f\left(\frac{x}{\sqrt{k}}\right) \geq \frac{1}{4}$. Applying Lemma 19 with $f, D = 4$ yields

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 168\sqrt{k}c. \tag{93}$$

□

Lemma 22. *Let $f = \log(1 + e^x)$ for every $x \in \mathbb{R}$ and let $c > 0$. Then, there is $k_0 > 0$ such that for every $k \geq k_0$ and for every $0 \leq x \leq R$*

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 3\frac{\log\left(2e^{cR}\right)}{\log(2)}\sqrt{k}c.$$

Proof. Let $0 \leq x \leq R$. Applying Lemma 17 with $k = c = 1$ yields $x_{1,1}$ such that $f(-x_{1,1}) = x_{1,1}^2$. We now bound $x_{1,1}$. Calculation shows that

$$f\left(-\sqrt{\ln(1.2)}\right) > \left(\sqrt{\ln(1.2)}\right)^2.$$

Plugging $x = \sqrt{\ln(1.2)}, k = 1, c = 1$ in Lemma 17**(ii)** yields

$$x_1 \geq \sqrt{\ln(1.2)}. \tag{94}$$

By applying Lemma 18 with f we have

$$\frac{f(x) + x^2}{f(-x) + x^2} \leq \max\left\{2, \frac{2}{x_{1,1}^2}\right\} \leq 11, \tag{95}$$

where the last inequality is by (94).

For every $c, k > 0$, since $x \leq R$ and f is increasing we have that

$$f(cx) \leq f(cR), \tag{96}$$

furthermore, since $x \geq 0$ and f is increasing we have that

$$f\left(\frac{x}{\sqrt{k}}\right) \geq \log(2). \tag{97}$$

We have that

$$\frac{f(cx)}{f\left(\frac{x}{\sqrt{k}}\right)} \leq \frac{f(cR)}{\log(2)} \tag{98}$$

$$= \frac{\log\left(1 + e^{cR}\right)}{\log(2)} \tag{99}$$

$$\leq \frac{\log\left(2e^{cR}\right)}{\log(2)}, \tag{100}$$

where (98) is by (96) and (97), (99) is by the definition of f and (100) holds since $Rc > 0$. Applying Lemma 19 with $f, D = \frac{\log\left(2e^{cR}\right)}{\log(2)}$ yields

$$\frac{f(cx) + \frac{x^2}{k}}{f(-cx) + \frac{x^2}{k}} \leq 3\frac{\log\left(2e^{cR}\right)}{\log(2)}\sqrt{k}c. \tag{101}$$

□

Theorem 23 (Intermediate Value Theorem)**.** *Let $a, b \in \mathbb{R}$ such that $a < b$ and let $f : [a, b] \to \mathbb{R}$ be a continuous function. Then for every u such that*

$$\min\{f(a), f(b)\} \leq u \leq \max\{f(a), f(b)\},$$

there is $c \in (a, b)$ such that $f(c) = u$.

Theorem 24 (Mean Value Theorem)**.** *Let $a, b \in \mathbb{R}$ such that $a < b$ and $f : [a, b] \to \mathbb{R}$ a continuous function on the closed interval $[a, b]$ and differentiable on the open interval (a, b). Then there is $c \in (a, b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Theorem 25 (Inverse of Strictly Monotone Function Theorem)**.** *Let $I \subseteq \mathbb{R}$. Let $f : I \to \mathbb{R}$ be strictly monotonic function. Let the image of f be J. Then f has an inverse function f^{-1} and*

- If f is strictly increasing then so is f^{-1}.
- If f is strictly decreasing then so is f^{-1}.