UNIVERSITY OF CALIFORNIA
SANTA CRUZ

TEMPERED BREGMAN DIVERGENCE FOR CONTINUOUS AND DISCRETE TIME MIRROR DESCENT AND ROBUST CLASSIFICATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Ehsan Amid

June 2020

The Dissertation of Ehsan Amid
is approved:

Professor Manfred K. Warmuth, Chair

Professor David Helmbold

Professor Abhradeep Guha Thakurta

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Tempered Bregman Divergence for Continuous and Discrete Time Mirror Descent and Robust Classification

by

Ehsan Amid

Bregman divergence is an important class of divergence functions in Machine Learning. Many well-known updates including gradient descent and (un)normalized exponentiated gradient are motivated by using a Bregman divergence as the inertia term. Moreover, Bregman divergence is used as a measure of progress for online algorithms as well as the training loss for classification models. In this thesis, we introduce a class of tempered Bregman divergences that, as special cases, includes many well-known distance measures such as squared Euclidean and relative entropy. We explore the tempered updates motivated by the new tempered Bregman divergence and develop theorems that allow us to unify these updates as gradient descent. We show the application of the reparameterized updates by proving regret bounds for the special case of reparameterized exponentiated gradient. Finally, we extend the notion of a matching loss to the new tempered Bregman divergence and develop bounded classification loss functions that are significantly more robust to noise and outliers.

تقدیم به مادر عزیزم

(To my dear Mother)

Acknowledgments

I would like to first and foremost thank my advisor, Manfred Warmuth, for all his help and support throughout these years. Apart from being an excellent advisor and a true inspiration, he has always been a caring friend.

I would like to also thank all the people with whom I have had the previlidge to collaborate. Especially, my colleagues at the Google Brain team, Rohan Anil and Tomer Koren, and Sriram Srinivasan at the UC Santa Cruz.

Many thanks to the committee members, David Helmbold, Abhradeep Guha Thakurta, and Wojciech Kotłowski, for their valuable feedback for preparing this thesis.

Last but not least, I am grateful to all the friends and family members for going through this journey with me.

Chapter 1

Introduction

Divergence functions are at the core of Machine Learning and Statistics. In general, divergence functions are used as a measure of discrepancy between a control object (i.e. model parameter, prediction, distribution, etc.) and its target value. In particular, divergence functions can be used as: 1) a regularization term in deriving parameter updates: the update thus balances the trade off between minimizing the objective function while remaining close to the old parameter. An important family of such updates that we study extensively in this thesis is called *mirror descent* [50], which includes gradient descent and exponentiated gradient updates [40] as special cases. 2) As a measure of progress in achieving worst-case regret bounds: classical regret bounds in online learning establish an upper-bound on the worst-case performance of an online algorithm compared to the best learner in hindsight. The progress of the algorithm towards an arbitrary learner is measured in terms of a divergence function. 3) As a way of defining well-behaved (convex) loss functions: the learner minimizes the discrepancy of the model prediction and the target. Classification is one of the main problems in this regard. In this thesis, we greatly advance the agenda in these aspects. We first review the previous work for each case in detail in the following.

1.1 Motivation and Previous Work

Given a parametric model and some initial values for the parameters, the goal of learning is to update the parameter values such that the model *fits* the data as accurately as possible. The goodness of the fit is generally measured in terms of a loss, commonly a divergence function, that punishes the discrepancy between the output of the model and the desired target dictated by the data. A good loss function should preferably take into account the structure of the problem. For instance, many different divergence functions are proposed for the cases where the model output and the target are probability distributions [19].

Usually, the learning proceeds in steps and the model maintains the best set of parameters at any instance. At each step, the parameters are updated to the minimizer of the loss plus an inertia term that keeps the new parameters close to the old ones. In general, the inertia term is another divergence function between the old and new parameters. Using different divergence functions for the parameters leads to different classes of parameter updates, as we will discuss later.

The main class of divergence functions include Csiszár f-divergence [22] (which includes the α-divergence [3] as a special case), optimal transport cost between probability distributions [66] and its approximation via entropic regularization [23], and Bregman divergence [16] (which includes the β-divergence [20] as a special case). Specifically, the f-divergence is defined via an expectation over the value of a convex function f applied to the ratio of two probability distributions. The most interesting parametric class of f-divergences is perhaps the Tsallis divergence [62]. The Tsallis divergence is motivated by the Tsallis entropy which is a generalization of the standard Boltzmann–Gibbs entropy with a temperature $\tau \in \mathbb{R}$. The standard entropy is recovered at $\tau = 1$. The Tsallis divergence is also closely related to the Rényi divergence [63].

The optimal transport cost, or more specifically the Wasserstein or Kantorovich–Rubinstein metric is defined as a distance function between two probability distributions on a given metric space [66]. Intuitively, the Wasserstein distance is the minimum amount of cost of moving one probability distribution into another. Since the calculation of the cost requires an optimization over all possible joint probability distributions, direct application of Wasserstein metric is infeasible in many cases. However, approximations of the distance in certain cases have been proposed using entropic regularization and Sinkhorn balancing [23].

The Bregman divergence is perhaps the most versatile divergence used in Machine Learning. It is defined as the difference between the growth of a strictly convex function and its linear approximator. The Bregman divergence includes many commonly used divergences such as the squared Euclidean, relative entropy, Burg (a.k.a. Itakura-Saito), inverse divergence, etc. The properties of the Bregman divergence have been studied throughout the years [52, 13, 15, 26, 65] and it has been successfully applied to clustering [12], matrix factorization [21], and density-ratio estimation [60]. In particular, the Bregman divergence can be used as the inertia term for parameter optimization, which leads to the class of mirror descent updates [50]. The main class of mirror descent updates are gradient descent and exponentiated gradient [40], which are motivated using the squared Euclidean and relative entropy divergence as inertia terms, respectively. For example, the classical Perceptron and Winnow algorithms are motivated using the identity and log links, respectively, when the loss is the hinge loss. The gradient descent and the exponentiated gradient updates have fundamentally different properties [67, 25] and for years the differences between the two have been studied extensively [40, 41, 51]. In particular, the *Hadamard problem* is a paradigmatic linear problem that has been used to highlight the difference between the two

updates [39, 67]. There has been recent attempts to approximately unify the two updates in different regions of the parameter space [30]. However, the unification holds only approximately and the transition between the two updates has been unclear.

In addition to efficient parameter updates, the Bregman divergence can also be used as a classification loss. Specifically, the relative entropy divergence paired with the softmax transfer function, called the *logistic loss*, is perhaps the most commonly used loss function for training classifiers. The main drawback of using the logistic loss for classification in noisy setting is that, due to the convexity of the loss, the effect of noisy examples can dominate the total loss and therefore, result in a poor classification margin [48]. There has been attempts to make the loss non-convex by bending down the loss for large-margin misclassified examples [28, 10]. However, these approaches fail to satisfy important properties for classification loss functions such as properness [72] and Bayes-risk consistency [43, 61].

1.2 Contributions

In this thesis, we provide a set of parametric mirror descent updates that interpolate between the gradient descent and the exponentiated gradient updates. For this, we first develop a tempered Bregman divergence that interpolates between the squared Euclidean and the relative entropy divergence. Our new tempered divergence also generalizes beyond the two main divergences and recovers other well-known divergences such as Burg and inverse. Next, we construct a reparameterization theorem that unifies the tempered updates (including the exponentiated gradient update) as gradient decent updates on a new set of parameters. We first show the exact equivalence in the continuous domain where the learning rate is infinitesimally small. We show that the tempered updates as well as their cor-

responding gradient descent equivalent converge to minimum norm solutions in certain settings. Next, we analyze the discretized updates for the exponentiated gradient updates and show that even after discretization, the known bounds for the exponentiated gradient updates still hold closely for the reparameterized approximations. Finally, we consider an application of the new tempered Bregman divergence for robust classification. That is, we extend the concept of a matching loss [35, 38] to the tempered divergence and show that the convexity holds when the temperature of the tempered softmax function matches the temperature of the Bregman divergence. However, we show that bounded non-convex loss functions that are significantly more robust to noise can be constructed by introducing a mismatch between the temperatures of the transfer function and the Bregman divergence. We show that the new *bi-tempered loss* can handle large amounts of label noise for training deep neural networks.

1.3 Outline of the Thesis

In Chapter 2, we review the preliminary topics for the remainder of the thesis, namely, convex duality, Bregman divergence, and two related concepts namely strong convexity and strong smoothness. We also discuss the dual of a convex function under an affine constraint. Using this, we revisit the notion of a matching loss and introduce a new formulation based on convex duality. In Chapter 3, we motivate the continuous-time mirror descent as the solution of a functional objective. Next, we show that in the continuous domain, a mirror descent update on a set of parameters can equivalently be written as an alternative mirror descent update on a different set of parameters given that a certain mapping exists between the parameters. In order to apply this trick on the exponentiated gradient updates, we first introduce the tempered Bregman divergence and show that the

different well-known Bregman divergences can be recovered as special cases. Next, we introduce the reparameterized form of the tempered updates as gradient decent updates and apply the results to the underdetermined linear regression problem. We show that under certain conditions, the solution converges to the minimum norm solutions w.r.t. the appropriate norm. In Chapter 4, we show that the discretization of the reparameterized exponentiated gradient updates closely mimics the original updates. Specifically, we show that the known bounds for the original updates for the Winnow and Hedge algorithms as well as the linear regression problem holds similarly for the reparameterized updates. In Chapter 5, we apply the tempered Bregman divergence to the classification problem and show that the tempered matching loss still induces a convex loss w.r.t the activations. Next, we introduce a mismatch between the temperatures of the tempered softmax function and the tempered Bregman divergence and show that for certain choice of temperatures, the loss becomes bounded (and therefore non-convex) and the transfer functions adopts a heavier-tail than the softmax function. We show that boundedness and tail-heaviness are important properties for handling small-margin and large-margin label noise, respectively. We also show that the new bi-tempered loss is proper and Bayes-risk consistent, even in the non-convex case. Finally, we conclude the thesis in Chapter 6 and provide a number of research directions as possible extensions.

1.4 Original Publications

A part of the material in Chapter 2 along with a discussion of the joint exponential family distributions for hidden variable models appeared as two conference papers in UAI 2020 [6] and AAAI 2020 [7], and its extensions will be submitted as a journal paper. The material in Chapter 3 is under review as a conference

paper in NeurIPS 2020 [8]. Chapter 4 appeared as a conference paper in COLT 2020 [9]. Finally, Chapter 5 appeared as two conference papers, one in AISTATS 2019 [10], and the other in NeurIPS 2019 [5].

1.5 Notation

We use $\odot$ and superscript $^{\odot}$ for element-wise product and element-wise power, respectively, and use $\oslash$ to denote element-wise division. We let $\boldsymbol{w}(t)$ denote the weight or parameter vector as a function of time t. Learning proceeds in steps. During step s, we start with weight vector $\boldsymbol{w}(s) = \boldsymbol{w}_s$ and go to $\boldsymbol{w}(s+1) = \boldsymbol{w}_{s+1}$ while processing a batch of examples. Similarly, we use $\boldsymbol{W}(t)$ for a continuous-time parameter matrix and denote $\boldsymbol{W}(s) = \boldsymbol{W}_s$. We also write the Jacobian of a function q evaluated at $\boldsymbol{v}$ as $\boldsymbol{J}_q(\boldsymbol{v})$ and use $\boldsymbol{H}_F(\boldsymbol{v})$ to denote the Hessian of a real-valued function F at $\boldsymbol{v}$. Also, $\mathbb{S}^d$ denotes the set of real-symmetric $d \times d$ matrices and Δ^n denotes the unit n-simplex.

In the continuous-time setting, the dot symbol $\dot{\boldsymbol{v}}(t) \doteq \frac{\partial}{\partial t}\boldsymbol{v}(t)$ denotes the time derivative of the vector $\boldsymbol{v}(t)$. Also, $\dot{f}(\boldsymbol{v}(t)) \doteq \frac{\partial}{\partial t} f(\boldsymbol{v}(t)) = \boldsymbol{J}_f(\boldsymbol{v}(t))^\top \dot{\boldsymbol{v}}(t)$ denotes the time derivative of the multivariate vector function f at $\boldsymbol{v}(t)$.

Chapter 2

Convex Duality, Bregman Divergence, and Matching Loss

In this chapter, we review the material that will be used throughout the dissertation. Specifically, we first discuss duality, Bregman divergence, and two related concepts, namely, strong convexity and strong smoothness. Next, we show how to calculate the dual of a convex function when the domain is constrained to an affine set. Using these tools, we review the notion of a *matching loss* which was initially introduced in [35, 38] by means of area under integrals. We generalize the matching loss in terms of convex duality to the cases where the domain is constrained to an affine set. For a more detailed discussion on convexity, the interested reader is referred to standard textbooks such as [37]. A part of the material in this chapter appeared as two conference papers in UAI 2020 [6] and AAAI 2020 [7]

2.1 Convex Duality

Let $F : \mathcal{D} \to \mathbb{R} \cup \{-\infty, +\infty\}$ be a proper, continuously-differentiable convex function defined on the convex domain $\mathcal{D} \subseteq \mathbb{R}^d$ taking values on the extended real line. The convex conjugate of F, denoted by $F^* : \mathcal{D}^* \to \mathbb{R} \cup \{-\infty, +\infty\}$, is defined in terms of the supremum

$$F^*(\boldsymbol{\theta}) = \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}} \{ \boldsymbol{\theta} \cdot \tilde{\boldsymbol{w}} - F(\tilde{\boldsymbol{w}}) \} .$$

Denoting $\boldsymbol{w} = \arg\sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}} \{ \boldsymbol{\theta} \cdot \tilde{\boldsymbol{w}} - F(\tilde{\boldsymbol{w}}) \}$, the following relations hold between the *dual variables* $(\boldsymbol{w}, \boldsymbol{\theta})$ and the *link functions* $f \doteq \nabla F$ and $f^* \doteq \nabla F^*$,

$$\boldsymbol{\theta} = f(\boldsymbol{w}) , \quad \boldsymbol{w} = f^*(\boldsymbol{\theta}) , \quad \text{and} \quad f^* = f^{-1} . \tag{2.1}$$

Additionally, by lower semi-continuity of F, we have

$$F(\boldsymbol{w}) = \sup_{\tilde{\boldsymbol{\theta}} \in \mathcal{D}^*} \{ \tilde{\boldsymbol{\theta}} \cdot \boldsymbol{w} - F^*(\tilde{\boldsymbol{\theta}}) \} = \boldsymbol{\theta} \cdot \boldsymbol{w} - F^*(\boldsymbol{\theta}) ,$$

where $\mathcal{D}^*$ is the domain of the dual function F^*. Later, we will discuss how the convex dual of F is affected when the domain of F is constrained to an affine set. We first review a closely related concept, namely, the Bregman divergence.

2.2 Bregman Divergence

Given a continuously-differentiable, strictly convex function F defined on the convex domain $\mathcal{D}$, the Bregman divergence [16] between $\tilde{\boldsymbol{w}}$, $\boldsymbol{w} \in \mathcal{D}$ induced by F

is defined as

$$D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = F(\tilde{\boldsymbol{w}}) - F(\boldsymbol{w}) - f(\boldsymbol{w}) \cdot (\tilde{\boldsymbol{w}} - \boldsymbol{w}) .$$

The Bregman divergence includes many well-known divergences commonly used in practice, namely, the squared Euclidean distance, relative entropy, Burg divergence, etc. Also, the Bregman divergence enjoys many favorable properties, making it an active topic of research for many decades [40, 11, 12, 52, 26, 21, 15, 65, 60, 13]. For instance, using Bregman divergence as an inertia term leads to a class of updates called *mirror descent* [50], which will be discussed thoroughly in the later chapters. The main properties of the Bregman divergence can be summarized as follows:

- **Non-negativity:** $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) \geq 0$ and $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = 0$ if and only if $\tilde{\boldsymbol{w}} = \boldsymbol{w}$.
- **Convexity:** the Bregman divergence $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w})$ is always convex in the first argument, but not necessarily in the second argument.
- **Asymmetry:** $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) \neq D_F(\boldsymbol{w}, \tilde{\boldsymbol{w}})$ in general.
- **Invariance to addition of affine function:** $D_{F+A}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w})$ where $A(\boldsymbol{w}) = \boldsymbol{a} \cdot \boldsymbol{w} + b$ for arbitrary $\boldsymbol{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.
- **Duality:** $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = D_{F^*}(f(\boldsymbol{w}), f(\tilde{\boldsymbol{w}})) = D_{F^*}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$ for the pairs of dual variables $(\boldsymbol{w}, \boldsymbol{\theta})$ and $(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{\theta}})$.
- **Derivatives:** the derivatives of $D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w})$ w.r.t. the first and second arguments yield

  $$\nabla_{\tilde{\boldsymbol{w}}} D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = f(\tilde{\boldsymbol{w}}) - f(\boldsymbol{w}) , \quad \text{and} \quad \nabla_{\boldsymbol{w}} D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = -\boldsymbol{H}_F(\boldsymbol{w}) (\tilde{\boldsymbol{w}} - \boldsymbol{w}) .$$

where $\boldsymbol{H}_F(\boldsymbol{w}) \doteq \nabla^2 F(\boldsymbol{w})$ denotes the Hessian of F, evaluated at $\boldsymbol{w}$.

The following lemmas are useful for combining Bregman divergences.

Lemma 1 (Forward Combining)**.** *Let $\{\boldsymbol{w}_i \in \mathcal{D}\}_{i=1}^k$ be a set of variables with corresponding dual variables $\{\boldsymbol{\theta}_i \in \mathcal{D}^*\}_{i=1}^k$. Let*

$$\boldsymbol{w}_{\text{opt}} = \arg\min_{\tilde{\boldsymbol{w}}} \sum_i \alpha_i \, D_F(\tilde{\boldsymbol{w}}, \, \boldsymbol{w}_i) \quad \textit{where} \quad \alpha_i \geq 0 \quad \textit{and} \quad \sum_i \alpha_i = 1 \, .$$

be the minimizer of the convex combination of forward Bregman divergences. We have

$$\boldsymbol{w}_{\text{opt}} = f^{-1}\Big(\sum_i \alpha_i f(\boldsymbol{w}_i)\Big) \, , \quad \textit{i.e.} \quad \boldsymbol{\theta}_{\text{opt}} = \sum_i \alpha_i \, \boldsymbol{\theta}_i \, ,$$

where $\boldsymbol{\theta}_{\text{opt}} = f^{-1}(\boldsymbol{w}_{\text{opt}})$ is the dual of $\boldsymbol{w}_{\text{opt}}$.

Proof. Setting the derivatives of the objective w.r.t. $\tilde{\boldsymbol{w}}$ to zero, we have

$$\sum_i \alpha_i \Big(f(\boldsymbol{w}_{\text{opt}}) - f(\boldsymbol{w}_i) \Big) = f(\boldsymbol{w}_{\text{opt}}) - \sum_i \alpha_i \, f(\boldsymbol{w}_i) = \boldsymbol{0} \, .$$

Applying the inverse function concludes the proof. □

Corollary 1 (Forward Combining Gap)**.** *Given in the conditions of Lemma 1, the following holds*

$$\begin{aligned} \sum_i \alpha_i \, D_F(\boldsymbol{w}, \boldsymbol{w}_i) \; &= \; D_F(\boldsymbol{w}, \boldsymbol{w}_{\text{opt}}) \; + \underbrace{\sum_i \alpha_i \, D_F(\boldsymbol{w}_{\text{opt}}, \boldsymbol{w}_i)}_{\text{optimum objective}} \\ &= \sum_i \alpha_i \, F^*(f(\boldsymbol{w}_i)) - F^*(f(\boldsymbol{w}_{\text{opt}})) \, , \end{aligned}$$

for any $\boldsymbol{w} \in \mathcal{D}$ where $\boldsymbol{w}_{\text{opt}} = f^{-1}\big(\sum_i \alpha_i f(\boldsymbol{w}_i)\big)$.

Lemma 2 (Backward Combining)**.** *Let* $\{\boldsymbol{w}_i \in \mathcal{D}\}_{i=1}^k$ *and* $\{\boldsymbol{\theta}_i \in \mathcal{D}^*\}_{i=1}^k$ *be the set of variables as in Lemma 1. Let*

$$\boldsymbol{w}_{\text{opt}} = \arg\min_{\tilde{\boldsymbol{w}}} \sum_i \alpha_i \, D_F(\boldsymbol{w}_i, \, \tilde{\boldsymbol{w}}) \quad \textit{where} \quad \alpha_i \geq 0 \quad \textit{and} \quad \sum_i \alpha_i = 1 \,,$$

be the minimizer of the convex combination of backward Bregman divergences. We have

$$\boldsymbol{w}_{\text{opt}} = \sum_i \alpha_i \, \boldsymbol{w}_i \,, \quad \textit{i.e.} \quad \boldsymbol{\theta}_{\text{opt}} = f\big(\sum_i \alpha_i \, f^{-1}(\boldsymbol{\theta}_i)\big) \,.$$

Corollary 2 (Backward Combining Gap)**.** *Given in the conditions of Lemma 2, the following holds*

$$\begin{aligned} \sum_i \alpha_i \, D_F(\boldsymbol{w}_i, \boldsymbol{w}) \; &= \underbrace{\sum_i \alpha_i \, D_F(\boldsymbol{w}_i, \boldsymbol{w}_{\text{opt}})}_{\text{optimum objective}} + D_F(\boldsymbol{w}_{\text{opt}}, \boldsymbol{w}) \\ &= \sum_i \alpha_i \, F(\boldsymbol{w}_i) - F(\boldsymbol{w}_{\text{opt}}) \,, \end{aligned}$$

for any $\boldsymbol{w} \in \mathcal{D}$ *where* $\boldsymbol{w}_{\text{opt}} = \sum_i \alpha_i \boldsymbol{w}_i$*.*

2.3 Strong Convexity and Smoothness

The following material for strong convexity and strong smoothness are adopted from [58]. The proofs are omitted for conciseness.

Definition 1 (σ-Strong Convexity)**.** *A continuous function* F *is* σ*-strongly convex w.r.t. the norm* $\|\cdot\|$ *over the convex set* $\mathcal{S}$ *if* $\mathcal{S}$ *is contained in the domain of* F *and the following inequality holds for any* $\tilde{\boldsymbol{w}}, \, \boldsymbol{w} \in \mathcal{S}$*,*

$$F(\tilde{\boldsymbol{w}}) \geq F(\boldsymbol{w}) + \nabla F(\boldsymbol{w}) \cdot (\tilde{\boldsymbol{w}} - \boldsymbol{w}) + \frac{\sigma}{2} \|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|^2 \,.$$

Lemma 3. *Assume F is twice differentiable. Then F is σ-strongly convex if*

$$\tilde{\boldsymbol{w}}^\top \boldsymbol{H}_F(\boldsymbol{w})\, \tilde{\boldsymbol{w}} \geq \sigma \|\tilde{\boldsymbol{w}}\|^2, \ \ \forall \tilde{\boldsymbol{w}}, \boldsymbol{w} \in \mathcal{S}\,.$$

Lemma 4. *Let F be a σ-strongly convex differentiable function over the non-empty convex set $\mathcal{S}$. For all $\tilde{\boldsymbol{w}}$, $\boldsymbol{w} \in \mathcal{S}$, we have*

$$\frac{\sigma}{2} \|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|^2 \ \leq \ D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w})\,.$$

Definition 2 (σ-Strong Smoothness)**.** *A differentiable function F is σ-strongly smooth w.r.t. the norm $\|\cdot\|$ if*

$$D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) \leq \frac{\sigma}{2} \|\tilde{\boldsymbol{w}} - \boldsymbol{w}\|^2\,.$$

Lemma 5. *Let F be a closed and convex function. Then F is σ-strongly convex w.r.t. the $\|\cdot\|$ norm if and only if F^*, the dual of F, is $\frac{1}{\sigma}$-strongly smooth w.r.t. the dual norm $\|\cdot\|_*$.*

2.4 Dual of a Constrained Convex Function

We now consider the dual of the convex function $F(\boldsymbol{w})$ where the domain is restricted to an affine set. The affine constraints appear naturally in many cases such as classification where the the weight vector $\boldsymbol{w}$ corresponds to the class probabilities and therefore, lies on the unit simplex. Affine constraints also appear in online learning where $\boldsymbol{w}$ corresponds to a distribution over the experts [46].

Theorem 1. *Let F be a differentiable proper convex function defined on the convex domain $\mathcal{D}$ with convex dual function F^*. Suppose the convex set $\mathcal{D}_{\text{aff}} \doteq \mathcal{D} \cap \{\tilde{\boldsymbol{w}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d\}$ is non-empty. Then, function F restricted to $\mathcal{D}_{\text{aff}}$ induces an alternate*

dual function $\check{F}^*$ *such that the new dual variable* $\check{\boldsymbol{\theta}}$ *corresponding to* $\boldsymbol{w} \in \mathcal{D}_{\text{aff}}$ *satisfies* $\check{\boldsymbol{\theta}} = \boldsymbol{P}\,\boldsymbol{\theta}$ *where* $\boldsymbol{P} \doteq \boldsymbol{I} - \frac{\boldsymbol{c}\,\boldsymbol{c}^\top}{\|\boldsymbol{c}\|^2}$ *is the projection matrix onto the affine set* $\{\tilde{\boldsymbol{w}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d\}$*. Additionally, the following relation holds between the constrained dual function* $\check{F}^*$ *and the unconstrained dual function* F^*

$$\check{F}^*(\check{\boldsymbol{\theta}}) = F^*(\boldsymbol{\theta}) - \lambda_{\text{proj}}\, d\,,$$

such that $\boldsymbol{\theta} = \check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}$ *holds.*

Proof. Let $\check{\boldsymbol{\theta}} \in \mathbb{R}^d$ be such that $\boldsymbol{P}\,\check{\boldsymbol{\theta}} = \check{\boldsymbol{\theta}}$. Using the definition of the convex dual, we have

$$\check{F}^*(\check{\boldsymbol{\theta}}) = \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D} \cap \{\boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d\}} \{\tilde{\boldsymbol{w}} \cdot \check{\boldsymbol{\theta}} - F(\tilde{\boldsymbol{w}})\} = \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}} \{\tilde{\boldsymbol{w}} \cdot \check{\boldsymbol{\theta}} - F(\tilde{\boldsymbol{w}}) + \lambda\,(\boldsymbol{c} \cdot \tilde{\boldsymbol{w}} - d)\}\,,$$

where in the second equality, we use a Lagrange multiplier $\lambda \in \mathbb{R}$ to enforce the affine constraint. Setting the derivatives w.r.t. $\tilde{\boldsymbol{w}}$ to zero, at the optimum $\boldsymbol{w}$, we have

$$f(\boldsymbol{w}) = \check{\boldsymbol{\theta}} + \lambda_{\text{proj}}(\check{\boldsymbol{\theta}})\,\boldsymbol{c} \quad \text{i.e.} \quad \boldsymbol{w} = f^{-1}(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}(\check{\boldsymbol{\theta}})\,\boldsymbol{c}) = f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}(\check{\boldsymbol{\theta}})\,\boldsymbol{c})\,, \quad (2.2)$$

where $\lambda_{\text{proj}}(\check{\boldsymbol{\theta}})$ denotes the value of the Lagrange multiplier for which the affine constraint $\boldsymbol{c} \cdot \boldsymbol{w} = d$ holds (notice the dependence on $\check{\boldsymbol{\theta}}$). Note that since $\boldsymbol{P}\,\check{\boldsymbol{\theta}} = \check{\boldsymbol{\theta}}$, the value of $\lambda_{\text{proj}}\,\boldsymbol{c} = (\boldsymbol{I} - \boldsymbol{P})\,f(\boldsymbol{w})$ corresponds to the component of $f(\boldsymbol{w})$ that is along the vector $\boldsymbol{c}$ and $\check{\boldsymbol{\theta}} = \boldsymbol{P}\,f(\boldsymbol{w})$ is the *projected gradient*, that is,

$$\begin{aligned} f(\boldsymbol{w}) \;=\; \boldsymbol{\theta} \;&=\; \boldsymbol{P}\,\boldsymbol{\theta} \;+\; (\boldsymbol{I} - \boldsymbol{P})\,\boldsymbol{\theta} \\ &=\; \check{\boldsymbol{\theta}} \;+\; \lambda_{\text{proj}}\,\boldsymbol{c}\,. \end{aligned}$$

Plugging in for $\boldsymbol{w}$, we have

$$\begin{aligned}
\check{F}^*(\check{\boldsymbol{\theta}}) &= \check{\boldsymbol{\theta}} \cdot f^{-1}(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - F(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) \\
&= (\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) \cdot f^{-1}(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - F(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - \lambda_{\text{proj}}\, \boldsymbol{c} \cdot f^{-1}(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) \\
&= F^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - \lambda_{\text{proj}}\, d\,. \qquad (2.3)
\end{aligned}$$

□

The following corollary connects the the unconstrained and constrained link functions.

Corollary 3. *The following relation holds between the unconstrained and constrained link functions:*

$$\check{f}^*(\check{\boldsymbol{\theta}}) = f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c})\,,$$

where λ_{proj} *is such that* $\boldsymbol{c} \cdot\ f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) = d$.

Proof. Taking the derivatives of both sides of (2.3) w.r.t. $\check{\boldsymbol{\theta}}$ yields

$$\begin{aligned}
\check{f}^*(\check{\boldsymbol{\theta}}) &= \frac{\partial(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c})}{\partial \check{\boldsymbol{\theta}}}\, f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - \frac{\partial \lambda_{\text{proj}}}{\partial \check{\boldsymbol{\theta}}}\, d \\
&= (\boldsymbol{I} + \frac{\partial \lambda_{\text{proj}}}{\partial \check{\boldsymbol{\theta}}}\, \boldsymbol{c}^\top)\, f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - \frac{\partial \lambda_{\text{proj}}}{\partial \check{\boldsymbol{\theta}}}\, d \\
&= f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c}) - \frac{\partial \lambda_{\text{proj}}}{\partial \check{\boldsymbol{\theta}}}\, (\boldsymbol{c} \cdot f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c})) - \frac{\partial \lambda_{\text{proj}}}{\partial \check{\boldsymbol{\theta}}}\, d \\
&= f^*(\check{\boldsymbol{\theta}} + \lambda_{\text{proj}}\, \boldsymbol{c})\,, \qquad (2.4)
\end{aligned}$$

□

Theorem 2. *The dual of the function* $\check{F}^*$ *corresponds to a convex function* $\check{F}$ *defined on the convex set* $\mathcal{D}_{\text{aff}}$ *such that* $\check{F}(\boldsymbol{w}) = F(\boldsymbol{w})$ *for any* $\boldsymbol{w} \in \mathcal{D}_{\text{aff}}$ *and* $\check{f}(\boldsymbol{w}) = \boldsymbol{P}\, f(\boldsymbol{w})$.

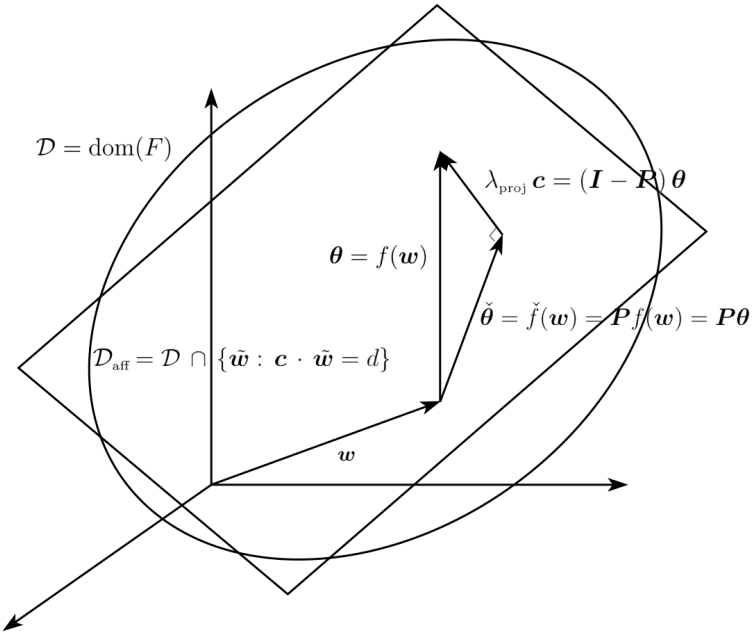


Figure 2.1: Projected gradient of a convex function with an affine constraint: the gradient of the unconstrained function $\boldsymbol{\theta}$ is decomposed into $\check{\boldsymbol{\theta}}$, which corresponds to the projected gradient that lies on the hyperplane plus $\lambda_{\mathrm{proj}}\boldsymbol{c}$, which is the term that is orthogonal to the hyperplane.

Proof. The dual of $\check{F}^*$ can be written as

$$\begin{aligned}
\check{F}(\boldsymbol{w}) &= \sup_{\tilde{\boldsymbol{\theta}}} \{\boldsymbol{w} \cdot \tilde{\boldsymbol{\theta}} - \check{F}^*(\tilde{\boldsymbol{\theta}})\} \\
&= \sup_{r \in \mathbb{R},\, \tilde{\boldsymbol{\theta}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{\theta}} = 0} \{\boldsymbol{w} \cdot (\tilde{\boldsymbol{\theta}} + r\, \boldsymbol{c}) - \check{F}^*(\tilde{\boldsymbol{\theta}} + r\, \boldsymbol{c})\} \\
&= \sup_{\tilde{\boldsymbol{\theta}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{\theta}} = 0} \{\boldsymbol{w} \cdot \tilde{\boldsymbol{\theta}} - \check{F}^*(\tilde{\boldsymbol{\theta}})\}\,.
\end{aligned}$$

Thus, any direction along $\boldsymbol{c}$ is vacuous and can be ignored. Setting the derivatives w.r.t. $\tilde{\boldsymbol{\theta}}$ to zero, and denoting the optimum by $\check{\boldsymbol{\theta}}$, we have

$$\boldsymbol{w} = \check{f}^*(\check{\boldsymbol{\theta}}) = f^*(\check{\boldsymbol{\theta}} + \lambda_{\mathrm{proj}}\, \boldsymbol{c})\,, \quad \text{i.e.} \quad \check{\boldsymbol{\theta}} = f(\boldsymbol{w}) - \lambda_{\mathrm{proj}}\, \boldsymbol{c}\,.$$

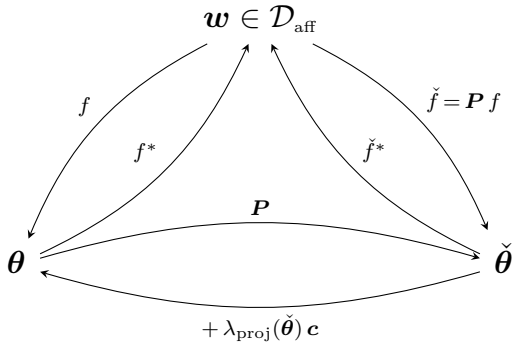


Figure 2.2: The relation between the primal variable $\boldsymbol{w}$ and its corresponding unconstrained dual $\boldsymbol{\theta}$ and constrained dual $\check{\boldsymbol{\theta}}$.

Plugging in the optimum value yields

$$\begin{aligned}\check{F}(\boldsymbol{w}) &= \boldsymbol{w} \cdot (f(\boldsymbol{w}) - \lambda_{\text{proj}}\, \boldsymbol{c}) - \check{F}^*(f(\boldsymbol{w})) + \lambda_{\text{proj}}\, d \\ &= \boldsymbol{w} \cdot f(\boldsymbol{w}) - \check{F}^*(f(\boldsymbol{w})) - \lambda_{\text{proj}}\, d + \lambda_{\text{proj}}\, d = F(\boldsymbol{w})\,.\end{aligned}$$

Finally, since $\text{dom}(\check{F}) = \mathcal{D}_{\text{aff}}$, we have $\check{f}(\boldsymbol{w}) = \boldsymbol{P}\, f(\boldsymbol{w})$ for $\boldsymbol{w} \in \mathcal{D}_{\text{aff}}$. □

Figure 2.1 pictorially shows the relation between the gradient and the projected gradient. In summary, the diagram in Figure 2.2 commutes between any $\boldsymbol{w} \in \mathcal{D}_{\text{aff}}$ and its unconstrained dual variable $\boldsymbol{\theta}$ and constrained dual variable $\check{\boldsymbol{\theta}}$.

An important property of the $\check{F}^*$ function and its link function $\check{f}^*$ is given in the following lemma.

Lemma 6. *The constrained dual function $\check{F}^*$ satisfies*

$$\check{F}^*(\check{\boldsymbol{\theta}} + r\, \boldsymbol{c}) = \check{F}^*(\check{\boldsymbol{\theta}}) + r\, d\,, \quad \textit{for any } r \in \mathbb{R}\,.$$

Proof. Using the definition of the dual function

$$\begin{aligned}\check{F}^*(\check{\boldsymbol{\theta}} + r\,\boldsymbol{c}) &= \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}_{\text{aff}}} \{\tilde{\boldsymbol{w}} \cdot (\tilde{\boldsymbol{\theta}} + r\,\boldsymbol{c}) - F(\tilde{\boldsymbol{w}})\} \\ &= \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}_{\text{aff}}} \{\tilde{\boldsymbol{w}} \cdot \tilde{\boldsymbol{\theta}} - F(\tilde{\boldsymbol{w}})\} + r\,d \\ &= \check{F}^*(\check{\boldsymbol{\theta}}) + r\,d\,.\end{aligned}$$

□

Corollary 4. *The constrained link function $\check{f}^*$ satisfies*

$$\check{f}^*(\check{\boldsymbol{\theta}} + r\,\boldsymbol{c}) = \check{f}^*(\check{\boldsymbol{\theta}})\,, \quad \textit{for any } r \in \mathbb{R}\,.$$

Proof. Taking the derivatives of both sides in Lemma 6 yields the result. □

Theorem 3 (Bregman projection onto the affine set)**.** *Let $\boldsymbol{w}' \in \mathcal{D}$ and let $\boldsymbol{w}^\star$ be the Bregman projection of $\boldsymbol{w}'$ onto the affine set $\mathcal{D}_{\text{aff}} = \mathcal{D} \cap \{\tilde{\boldsymbol{w}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d\}$,*

$$\boldsymbol{w}^\star = \operatorname*{arg\,min}_{\tilde{\boldsymbol{w}} \in \mathcal{D}_{\text{aff}}} D_F(\boldsymbol{w}', \tilde{\boldsymbol{w}})\,,$$

w.r.t. the convex function F. Then

$$\check{\boldsymbol{\theta}}^\star = \boldsymbol{P}\,f(\boldsymbol{w}')\,, \quad \textit{thus} \quad \boldsymbol{w}^\star = \check{f}^*(\boldsymbol{P}\,f(\boldsymbol{w}'))\textit{.}$$

Proof. Using a Lagrange multiplier λ to enforce the affine constraint $\boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d$, we have

$$\min_{\tilde{\boldsymbol{w}} \in \mathcal{D}_{\text{aff}}} D_F(\boldsymbol{w}', \tilde{\boldsymbol{w}}) = \min_{\tilde{\boldsymbol{w}} \in \mathcal{D}} D_F(\boldsymbol{w}', \tilde{\boldsymbol{w}}) + \lambda\,(\boldsymbol{c} \cdot \tilde{\boldsymbol{w}} - d)\,.$$

Setting the derivatives w.r.t. $\tilde{\boldsymbol{w}}$ to zero yields

$$f(\boldsymbol{w}') = f(\boldsymbol{w}^\star) - \lambda_{\text{proj}}\,\boldsymbol{c}\,,$$

where λ_{proj} is chosen such that $\boldsymbol{c} \cdot \boldsymbol{w}^\star = \boldsymbol{c} \cdot f^{-1}(f(\boldsymbol{w}') + \lambda_{\text{proj}}\, \boldsymbol{c}) = d$. The dual variable $\check{\boldsymbol{\theta}}^\star$ can be found by using the dual link function $\check{f}$, that is

$$\check{\boldsymbol{\theta}}^\star = \check{f}(\boldsymbol{w}^\star) = \boldsymbol{P}\, f(\boldsymbol{w}^\star) = \boldsymbol{P}\left(f(\boldsymbol{w}') + \lambda_{\text{proj}}\, \boldsymbol{c}\right) = \boldsymbol{P}\, f(\boldsymbol{w}')\,,$$

where we use the fact that $\boldsymbol{P}\, \boldsymbol{c} = \boldsymbol{0}$. The second claim follows immediately by applying the dual link $\check{f}^*$. □

Proposition 1. *The Bregman divergence induced by the convex function $\check{F}$ on the affine set $\operatorname{dom}(\check{F}) = \operatorname{dom}(F) \cap \{\tilde{\boldsymbol{w}} : \boldsymbol{c} \cdot \tilde{\boldsymbol{w}} = d\}$ is equivalent to the Bregman divergence induced by F.*

Proof. Given $\boldsymbol{w}, \boldsymbol{w}' \in \operatorname{dom}(\check{F})$, we have

$$\begin{aligned} D_{\check{F}}(\boldsymbol{w}, \boldsymbol{w}') &= \check{F}(\boldsymbol{w}) - \check{F}(\boldsymbol{w}') - \check{f}(\boldsymbol{w}') \cdot (\boldsymbol{w} - \boldsymbol{w}') \\ &= F(\boldsymbol{w}) - F(\boldsymbol{w}') - \boldsymbol{P}\, f(\boldsymbol{w}') \cdot (\boldsymbol{w} - \boldsymbol{w}') \\ &= F(\boldsymbol{w}) - F(\boldsymbol{w}') - f(\boldsymbol{w}') \cdot (\boldsymbol{P}\, \boldsymbol{w} - \boldsymbol{P}\, \boldsymbol{w}') \\ &= F(\boldsymbol{w}) - F(\boldsymbol{w}') - f(\boldsymbol{w}') \cdot (\boldsymbol{w} - \boldsymbol{w}') = D_F(\boldsymbol{w}, \boldsymbol{w}')\,, \end{aligned}$$

where we use the fact that $\boldsymbol{P}\, \boldsymbol{w} = \boldsymbol{w} - \frac{d}{\|\boldsymbol{c}\|^2}\boldsymbol{c}$ for any $\boldsymbol{w} \in \operatorname{dom}(\check{F})$. □

Recall that every Bregman divergence is equal to its dual Bregman divergence by flipping the arguments, i.e. $D_F(\boldsymbol{w}, \boldsymbol{w}') = D_{F^*}(\boldsymbol{\theta}', \boldsymbol{\theta})$ where $(\boldsymbol{w}, \boldsymbol{\theta})$ and $(\boldsymbol{w}', \boldsymbol{\theta}')$ are dual variables. The following corollary shows that the equality still holds under an affine constraint.

Corollary 5 (Bregman Duality Under an Affine Constraint)**.** *The following equality holds*

$$D_F(\boldsymbol{w}, \boldsymbol{w}') = D_{F^*}(\boldsymbol{\theta}', \boldsymbol{\theta}) = D_{\check{F}^*}(\check{\boldsymbol{\theta}}', \check{\boldsymbol{\theta}})\,,$$

where $\boldsymbol{\theta}$ ($\boldsymbol{\theta}'$) and $\check{\boldsymbol{\theta}}$ ($\check{\boldsymbol{\theta}}'$) are the unconstrained and constrained dual variables corresponding to $\boldsymbol{w}$ ($\boldsymbol{w}'$), respectively.

Proof. The proof immediately follows from Proposition 1. □

Example 1 (Negative Entropy Function)**.** *To exemplify the results of this section, consider the negative entropy function defined over the set of discrete non-negative measures $\boldsymbol{w} \in \mathbb{R}^d_{\geq 0}$, given by*

$$F_{\text{ent}}(\boldsymbol{w}) = \sum_i (w_i \log w_i - w_i)\,, \quad f_{\text{ent}}(\boldsymbol{w}) = \log \boldsymbol{w}\,.$$

The unconstrained dual of F_{ent} is given by

$$F^*_{\text{ent}}(\boldsymbol{\theta}) = \sum_i \exp \theta_i\,, \quad f^*_{\text{ent}}(\boldsymbol{\theta}) = f^{-1}_{\text{ent}}(\boldsymbol{\theta}) = \exp \boldsymbol{\theta} = \boldsymbol{w}\,.$$

Restricting the domain of F_{ent} to the set of non-negative probability measures $\boldsymbol{w} \in \mathbb{R}^d_{\geq 0}$, $\sum_i \boldsymbol{w}_i = \mathbf{1} \cdot \boldsymbol{w} = 1$ yields

$$\check{F}^*_{\text{ent}}(\check{\boldsymbol{\theta}}) = \log \sum_i \exp \check{\theta}_i\,, \quad \check{f}^*_{\text{ent}}(\check{\boldsymbol{\theta}}) = \frac{\exp \check{\boldsymbol{\theta}}}{\sum_i \exp \check{\theta}_i} = \boldsymbol{w}\,,$$

where $\check{\boldsymbol{\theta}} = \check{f}_{\text{ent}}(\boldsymbol{w}) = \boldsymbol{P}\, f_{\text{ent}}(\boldsymbol{w}) = \log \boldsymbol{w} - \frac{1}{d} \sum_i \log w_i$. Plugging back $\boldsymbol{w}$ into the link function yields

$$\log \frac{\exp \check{\boldsymbol{\theta}}}{\sum_i \exp \check{\theta}_i} - \frac{1}{d} \sum_j \log \frac{\exp \check{\theta}_j}{\sum_i \exp \check{\theta}_i} = \check{\boldsymbol{\theta}} - \underbrace{\frac{1}{d} \sum_j \check{\theta}_j}_{=0} = \check{\boldsymbol{\theta}}\,,$$

which confirms that $\check{f}_{\text{ent}}$ is in fact the inverse of $\check{f}^*_{\text{ent}}$. The converse relation is

trivial to verify. Moreover,

$$\begin{aligned}\lambda_{\text{opt}} &= (\boldsymbol{I} - \boldsymbol{P})\, f_{\text{ent}}(\boldsymbol{w}) \\ &= \frac{1}{d} \sum_i \log \frac{\exp \check{\theta}_i}{\sum_j \exp \check{\theta}_j} = \frac{1}{d} \underbrace{\sum_i \check{\theta}_i}_{=0} - \log \sum_j \exp \check{\theta}_j = -\log \sum_j \exp \check{\theta}_j \,.\end{aligned}$$

By Corollary 3, we have

$$f^*_{\text{ent}}(\check{\boldsymbol{\theta}} + \lambda_{\text{opt}}\, \mathbf{1}) = \exp(\check{\boldsymbol{\theta}} - \log \sum_j \exp \check{\theta}_j) = \frac{\exp \check{\boldsymbol{\theta}}}{\sum_j \exp \check{\theta}_j} = \check{f}^*_{\text{ent}}(\check{\boldsymbol{\theta}}) \,.$$

2.5 Matching Loss

Classification is one of the core problems in Machine Learning. A classifier consists of a model which, given an input $\boldsymbol{x}$, produces a probability distribution $\hat{\boldsymbol{y}}$ over the classes at the output. In general, the output probabilities are generated by applying a *transfer function* $s : \mathbb{R}^k \to \Delta^{k-1}$ on the output activations $\hat{\boldsymbol{a}} \in \mathbb{R}^k$, that is $\hat{\boldsymbol{y}} = s(\hat{\boldsymbol{a}})$. Usually, the class with the highest probability is chosen as the predicted class for the input.

In order to train a classifier, we need to define a notion of *loss* which establishes a measure of discrepancy between the output probabilities $\hat{\boldsymbol{y}}$ and the given target class probabilities (i.e. labels) $\boldsymbol{y}$. The classification loss functions are of primary interest in Machine Learning. The most basic loss function for classification consists of 0/1-loss which assigns a unit of loss when the instance is misclassified. Minimizing the 0/1-loss directly is a challenging task and is shown to be NP-hard [14]. The main challenge in minimizing the 0/1-loss is its non-convexity. Therefore, many convex loss functions have been proposed as convex surrogates. A well-known example is the logistic loss which consists of the relative entropy

divergence applied to the softmax function on the activations (which induces the class probabilities).

In this section, we recall the notion of a *matching loss* [35, 38, 17, 55]. It arises as a natural way of defining a loss function over activations $\hat{\boldsymbol{a}} \in \mathbb{R}^k$, by first mapping them to a probability distribution over class labels using a transfer function s, and then computing a divergence D_F between this distribution and the correct target labels. The idea behind the following definition is to "match" the transfer function and the divergence via duality.[1]

Definition 3 (Matching Loss)**.** *Let $F : \mathcal{D} \to \mathbb{R} \cap \{-\infty, +\infty\}$ be a continuously-differentiable, strictly convex function such that $\Delta^{k-1} \subseteq \mathcal{D}$ and let $s : \mathbb{R}^k \to \Delta^{k-1}$ be a transfer function such that $\hat{\boldsymbol{y}} = s(\hat{\boldsymbol{a}})$ denotes the predicted probability distribution based on the activations $\hat{\boldsymbol{a}}$. Then the loss function*

$$L_F(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) \doteq D_F(\boldsymbol{y}, s(\hat{\boldsymbol{a}}))\,,$$

is called the matching loss for s, if $s = \check{f}^* = \nabla \check{F}^*$.

Note that $\check{f}^*$ is no longer one-to-one since $\check{f}^*(\hat{\boldsymbol{a}} + \mathbb{R}\,\mathbf{1}) = \check{f}^*(\hat{\boldsymbol{a}})$ (Corollary 4). However, w.l.o.g. we can constrain the domain of the function to $\hat{\boldsymbol{a}} \in \operatorname{dom}(\check{f}^*) \cap \{\boldsymbol{a}' \in \mathbb{R}^k \mid \boldsymbol{a}' \cdot \mathbf{1} = 0\}$ to obtain a one-to-one mapping. The matching loss is useful due to the following property.

Proposition 2. *The matching loss $L_F(\hat{\boldsymbol{a}} \mid \boldsymbol{y})$ is convex w.r.t. the activations $\hat{\boldsymbol{a}} \in \operatorname{dom}(\check{f}^*) \cap \{\boldsymbol{a}' \in \mathbb{R}^k \mid \boldsymbol{a}' \cdot \mathbf{1} = 0\}$.*

Proof. Note that $\check{F}^*$ is a strictly convex function and the following relation holds

[1] Originally in [35, 38], the matching loss was defined as a simple integral over the transfer function $s = f^{-1}$: $L_F(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) = \int_{s^{-1}(\boldsymbol{y})}^{\hat{\boldsymbol{a}}} (s(\boldsymbol{z}) - \boldsymbol{y}) \cdot \mathrm{d}\,\boldsymbol{z}$. Our new duality based definition handles additional linear constraints.

between the divergences induced by F and $\check{F}^*$ (Corollary 5):

$$D_F(\boldsymbol{y}, \hat{\boldsymbol{y}}) = D_{\check{F}^*}\Big((\check{f}^*)^{-1}(\hat{\boldsymbol{y}}),\, (\check{f}^*)^{-1}(\boldsymbol{y})\Big). \tag{2.5}$$

Thus for any $\hat{\boldsymbol{a}}$ in the range of $(\check{f}^*)^{-1}$,

$$D_F\Big(\boldsymbol{y}, \check{f}^*(\hat{\boldsymbol{a}})\Big) = D_{\check{F}^*}\Big(\hat{\boldsymbol{a}}, (\check{f}^*)^{-1}(\boldsymbol{y})\Big).$$

The claim now follows from the convexity of $D_{\check{F}^*}$ w.r.t. its first argument. □

The original motivating example for the matching loss was the logistic loss [35, 38]. It can be obtained as the matching loss for the softmax function (see Example 1)

$$\hat{y}_i = [\check{f}^*(\hat{\boldsymbol{a}})]_i = \frac{\exp(\hat{a}_i)}{\sum_{j=1}^k \exp(\hat{a}_j)},$$

which corresponds to the relative entropy (KL) divergence

$$\begin{aligned} L_F(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) = D_F\Big(\boldsymbol{y}, \check{f}^*(\hat{\boldsymbol{a}})\Big) &= \sum_{i=1}^k y_i \left(\log y_i - \log \hat{y}_i\right) \\ &= \sum_{i=1}^k \Big(y_i \, \log y_i - y_i \, \hat{a}_i)\Big) + \log\Big(\sum_{i=1}^k \exp(\hat{a}_i)\Big)\,, \end{aligned}$$

induced from the negative entropy function $F(\boldsymbol{y}) = \sum_{i=1}^k (y_i \log y_i - y_i)$. In Chapter 3, we define a family of convex functions F_τ parameterized by a temperature $\tau \in \mathbb{R}$. In Chapter 5, we show that the matching loss $L_{F_\tau}(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) = D_{F_\tau}\Big(\boldsymbol{y}, \check{f}^*_\tau(\hat{\boldsymbol{a}})\Big)$ for the link function $\check{f}^*_\tau$ of $\check{F}^*_\tau$ is convex in the activations $\hat{\boldsymbol{a}}$. However, by letting the temperature τ_2 of $\check{f}^*_{\tau_2}$ be larger than the temperature τ_1 of F_{τ_1}, we construct bounded non-convex losses with heavy-tailed transfer functions. As we will see, boundedness of the loss and tail-heaviness of the transfer function are useful properties for handling large-margin and small-margin label noise, respectively, for training deep neural networks.

Chapter 3

Reparameterizing Mirror Descent as Gradient Descent

In this chapter, we revisit the continuous-time mirror descent (CMD) [69, 53] and provide a new interpretation based on minimizing a functional form of Bregman divergence plus loss. Next, we show that different discritizations can be achieved by direct Euler approximation of CMD or its natural gradient [4] form. We then introduce the main result of the chapter which allows reparameterizing one CMD as another. Using a proposed tempered Bregman divergence, we show that many well-known CMD updates can be unified as gradient descent (GD). Finally, we analyze the tempered updates and their reparameterized forms for the linear regression problem. The material in chapter currently is under review as a conference submission [8].

3.1 Introduction

Mirror Descent (MD) [50, 40] refers to a family of updates which transform the parameters $\boldsymbol{w} \in \mathcal{D}$ from a convex domain $\mathcal{D} \subseteq \mathbb{R}^d$ via a *link* function (a.k.a.

mirror map) $f : \mathcal{D} \to \mathbb{R}^d$ before applying the descent step. The *continuous-time mirror descent* update, which can be seen as the limit case of (discrete-time) MD, corresponds to the solution of the following ordinary differential equation (ODE) [50, 69, 53]:

$$\dot{f}(\boldsymbol{w}(t)) = -\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \,, \tag{3.1}$$

where $\dot{f} \doteq \frac{\partial f}{\partial t}$ is the time derivative of the link function. The main link functions investigated in the past are $f(\boldsymbol{w}) = \boldsymbol{w}$ and $f(\boldsymbol{w}) = \log(\boldsymbol{w})$ leading to the gradient descent (GD) and the unnormalized exponentiated gradient (EGU) family of updates[1]. These two link functions are associated with the squared Euclidean and the relative entropy divergences, respectively. For example, the classical Perceptron and Winnow algorithms are motivated using the identity and log links, respectively, when the loss is the hinge loss. A number of papers discuss the difference between the two updates [40, 41, 51, 30] and their rotational invariance properties have been explored in [70]. In particular, the *Hadamard problem* is a paradigmatic linear problem that shows that EGU can converge dramatically faster than GD when the instances are dense and the target weight vector is sparse [39, 67]. This property is linked to the strong-convexity of the relative entropy w.r.t. the L_1-norm[2] [59].

In this chapter, we introduce a family of *tempered* updates (parameterized by a *temperature* $\tau \in \mathbb{R}$) that interpolate between GD (with $\tau = 0$) and EGU (with $\tau = 1$) while covering a wider class of updates such as those motivated using the Burg and inverse divergences. Next, we show that all these updates can be unified as GD updates via a simple reparameterization. For this, we first carefully analyze the CMD updates and discuss the relation between the primal and dual updates.

[1]The normalized version is called EG and the two-sided version EGU$^{\pm}$. More about this later.

[2]Whereas the squared Euclidean divergence is strongly-convex w.r.t. the L_2-norm.

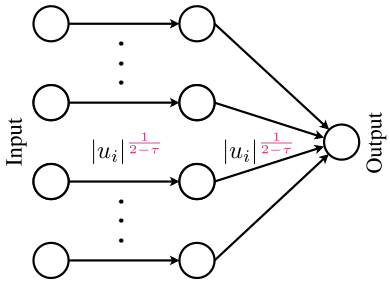


Figure 3.1: A reparameterized linear neuron where $w_i = |u_i|^{\frac{2}{2-\tau}}$ as a two-layer sparse network: value of $\tau = 0$ reduces to GD while $\tau = 1$ simulates the EGU update.

We also derive the constrained updates (e.g. when $\boldsymbol{w}$ lies on the simplex). Using these results, we show an equivalence relation between a CMD update and its reparameterization, given that the reparameterization mapping satisfies a certain condition. Finally, we apply our findings and provide reparameterizations of the new tempered updates as GD updates and show that for the underdetermined (a.k.a. over-parameterized) linear regression problem, these updates converge to the minimum $\mathrm{L}_{2-\tau}$-norm solutions for values of $\tau \in [0, 1]$. Figure 3.1 shows an example of a reparameterized linear neuron with temperature τ for this problem. We also provide partial results for the matrix case that generalize the results in [31]. We conclude the chapter with experiments and post a number of open problems for future research directions.

3.2 Continuous-time Mirror Descent

Recall that for a strictly convex, continuously-differentiable function $F : \mathcal{D} \to \mathbb{R} \cap \{-\infty, +\infty\}$ with convex domain $\mathcal{D} \subseteq \mathbb{R}^d$, the Bregman divergence between

$\tilde{\boldsymbol{w}}, \boldsymbol{w} \in \mathcal{D}$ is defined as

$$D_F(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = F(\tilde{\boldsymbol{w}}) - F(\boldsymbol{w}) - f(\boldsymbol{w}) \cdot (\tilde{\boldsymbol{w}} - \boldsymbol{w}) \,,$$

where $f \doteq \frac{\partial F}{\partial \boldsymbol{w}} = \nabla_{\boldsymbol{w}} F$ denotes the gradient[3] of F, sometimes called the *link function*. Trading off the divergence to the last parameter $\boldsymbol{w}_s$ with the current loss lets us motivate the iterative *mirror descent* (MD) updates [50, 40]:

$$\boldsymbol{w}_{s+1} = \arg\min_{\boldsymbol{w}} \, {}^1\!/\!_\eta \, D_F(\boldsymbol{w}, \boldsymbol{w}_s) + L(\boldsymbol{w}) \,, \tag{3.2}$$

where $\eta > 0$ is often called the learning rate. Solving for $\boldsymbol{w}_{s+1}$ yields the so-called *prox* or *implicit update* [56]:

$$f(\boldsymbol{w}_{s+1}) = f(\boldsymbol{w}_s) - \eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_{s+1}) \,. \tag{3.3}$$

This update is typically approximated by the following *explicit* update that uses the gradient at the old parameter $\boldsymbol{w}_s$ instead:

$$f(\boldsymbol{w}_{s+1}) = f(\boldsymbol{w}_s) - \eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s) \,. \tag{3.4}$$

We now show that the CMD update (3.1) can be motivated similarly by replacing the Bregman divergence in the minimization problem (3.2) with a "Bregman momentum" which quantifies the rate of change in the value of Bregman divergence as $\boldsymbol{w}(t)$ varies over time: For the convex function F, we define the *Bregman momentum* between $\boldsymbol{w}(t), \boldsymbol{w}_0 \in \mathcal{D}$ as the time differential of the Bregman divergence

[3]The gradient of a scalar function is a special case of a Jacobian, and should therefore be treated as a row vector. However, in this chapter we write the gradients of scalar functions as column vectors.

induced by F,

$$\begin{aligned}\dot{D}_F(\boldsymbol{w}(t), \boldsymbol{w}_0) &= \dot{F}(\boldsymbol{w}(t)) - f(\boldsymbol{w}_0) \cdot \dot{\boldsymbol{w}}(t) \\ &= \Big(f(\boldsymbol{w}(t)) - f(\boldsymbol{w}_0)\Big) \cdot \dot{\boldsymbol{w}}(t)\,.\end{aligned}$$

Proposition 3. *The CMD update*

$$\dot{f}\Big(\boldsymbol{w}(t)\Big) = -\eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))\,, \;\; \textit{with } \boldsymbol{w}(s) = \boldsymbol{w}_s,$$

is the solution of the following functional:

$$\min_{\boldsymbol{w}(t)} \Big\{ {}^1\!/_\eta\, \dot{D}_F(\boldsymbol{w}(t), \boldsymbol{w}_s) + L(\boldsymbol{w}(t)) \Big\}\,. \tag{3.5}$$

Proof. Setting the derivatives w.r.t. $\boldsymbol{w}(t)$ to zero, we have

$$\begin{aligned}&\frac{\partial}{\partial \boldsymbol{w}(t)} \Big(\Big(f(\boldsymbol{w}(t)) - f(\boldsymbol{w}_s)\Big) \cdot \dot{\boldsymbol{w}}(t) + \eta\, L(\boldsymbol{w}(t)) \Big) \\ &\quad = \boldsymbol{H}_F(\boldsymbol{w})\, \dot{\boldsymbol{w}}(t) + \frac{\partial \dot{\boldsymbol{w}}(t)}{\partial \boldsymbol{w}(t)} \Big(f(\boldsymbol{w}(t)) - f(\boldsymbol{w}_s)\Big) + \eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \\ &\quad = \dot{f}\Big(\boldsymbol{w}(t)\Big) + \eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) = \mathbf{0}\,,\end{aligned}$$

where we use the fact that $\boldsymbol{w}(t)$ and $\dot{\boldsymbol{w}}(t)$ are independent variables, therefore $\frac{\partial \dot{\boldsymbol{w}}(t)}{\partial \boldsymbol{w}(t)} = \mathbf{0}$. □

Note that the implicit update (3.3) and the explicit update (3.4) can both be realized as the backward and the forward Euler approximations of (3.1), respectively.

We can provide an alternative definition of Bregman momentum in terms of the dual of F function. If $F^*(\boldsymbol{\theta}) = \sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}} \Big(\boldsymbol{\theta} \cdot \tilde{\boldsymbol{w}} - F(\tilde{\boldsymbol{w}})\Big)$ denotes the Fenchel dual of F and $\boldsymbol{w} = \arg\sup_{\tilde{\boldsymbol{w}} \in \mathcal{D}} (\boldsymbol{\theta} \cdot \tilde{\boldsymbol{w}} - F(\tilde{\boldsymbol{w}}))$, then the following relation holds

between the pair of dual variables $(\boldsymbol{w}, \boldsymbol{\theta})$:

$$\boldsymbol{w} = f^*(\boldsymbol{\theta})\,, \quad \boldsymbol{\theta} = f(\boldsymbol{w})\,, \quad \text{and} \quad f^* = f^{-1}\,. \tag{3.6}$$

Taking the derivative of $\boldsymbol{w}(t)$ and $\boldsymbol{\theta}(t)$ w.r.t. t yields:

$$\begin{aligned} \dot{\boldsymbol{w}}(t) &= \dot{f^*}\Big(\boldsymbol{\theta}(t)\Big) = \boldsymbol{H}_{F^*}\Big(\boldsymbol{\theta}(t)\Big)\,\dot{\boldsymbol{\theta}}(t)\,, \\ \dot{\boldsymbol{\theta}}(t) &= \dot{f}\Big(\boldsymbol{w}(t)\Big) = \boldsymbol{H}_{F}\Big(\boldsymbol{w}(t)\Big)\,\dot{\boldsymbol{w}}(t)\,. \end{aligned} \tag{3.7}$$

This pairing lets us rewrite the Bregman momentum in its dual form. The dual form of Bregman momentum can be obtained by first forming the dual Bregman divergence in terms of the dual variables $\boldsymbol{\theta}(t)$ and $\boldsymbol{\theta}_0$ and taking the time derivative, that is,

$$\begin{aligned} &\dot{D_F}(\boldsymbol{w}(t), \boldsymbol{w}_0) = \dot{D_{F^*}}(\boldsymbol{\theta}_0, \boldsymbol{\theta}(t)) \\ &= \frac{\partial}{\partial t}\Big(F^*(\boldsymbol{\theta}_0) - F^*(\boldsymbol{\theta}(t)) - f^*(\boldsymbol{\theta}(t)) \cdot \Big(\boldsymbol{\theta}_0 - \boldsymbol{\theta}(t)\Big)\Big) \\ &= -\dot{F^*}(\boldsymbol{\theta}(t)) + f^*(\boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t) + (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0)^\top \boldsymbol{H}_{F^*}(\boldsymbol{\theta}(t))\,\dot{\boldsymbol{\theta}}(t) \\ &= \Big(\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0\Big)^\top \boldsymbol{H}_{F^*}(\boldsymbol{\theta}(t))\,\dot{\boldsymbol{\theta}}(t)\,, \end{aligned}$$

where we use the fact that $\dot{F^*}(\boldsymbol{\theta}(t)) = f^*(\boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t)$.

$$\dot{D_F}(\boldsymbol{w}(t), \boldsymbol{w}_0) = \dot{D_{F^*}}(\boldsymbol{\theta}_0, \boldsymbol{\theta}(t)) = (\boldsymbol{\theta}(t) - \boldsymbol{\theta}_0)^\top \boldsymbol{H}_{F^*}(\boldsymbol{\theta}(t))\,\dot{\boldsymbol{\theta}}(t)\,. \tag{3.8}$$

Using (3.7), we can rewrite the CMD update (3.1) as

$$\dot{\boldsymbol{w}}(t) = -\eta\,\boldsymbol{H}_F^{-1}(\boldsymbol{w}(t))\,\nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))\,, \tag{3.9}$$

which corresponds to a natural gradient update [4] w.r.t. the Riemannian metric

$\boldsymbol{H}_F$. Using $\boldsymbol{H}_F(\boldsymbol{w}) = \boldsymbol{H}_{F^*}^{-1}(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \boldsymbol{H}_{F^*}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} L \circ f^*(\boldsymbol{\theta})$, the update can also be written equivalently in the dual domain $\boldsymbol{\theta}$ as a natural gradient update w.r.t. the Riemannian metric $\boldsymbol{H}_{F^*}$, or as a CMD with the link f^*:

$$\dot{\boldsymbol{\theta}}(t) = -\eta \, \boldsymbol{H}_{F^*}^{-1}(\boldsymbol{\theta}(t)) \, \nabla_{\boldsymbol{\theta}} L \circ f^*(\boldsymbol{\theta}(t)) \,, \tag{3.10}$$

$$\dot{f^*}(\boldsymbol{\theta}(t)) = -\eta \, \nabla_{\boldsymbol{\theta}} L \circ f^*(\boldsymbol{\theta}(t)) \,. \tag{3.11}$$

The above equivalence between the primal and dual versions of the natural gradient update can also be seen as a special form of the reparameterization method presented in the next section (Corollary 6).

The CMD update naturally generalizes to the case where there exists a number of constraints on the parameter $\boldsymbol{w}(t)$. Essentially, the gradient on the r.h.s. is replaced by a projected gradient.

Proposition 4. *The CMD update with the additional constraint* $\psi\Big(\boldsymbol{w}(t)\Big) = \boldsymbol{0}$ *for some function* $\psi : \mathbb{R}^d \to \mathbb{R}^m$ *s.t.* $\{\boldsymbol{w} \in \mathcal{D} | \, \psi\Big(\boldsymbol{w}(t)\Big) = \boldsymbol{0}\}$ *is non-empty, amounts to the projected gradient update*

$$\dot{f}\Big(\boldsymbol{w}(t)\Big) = -\eta \, \boldsymbol{P}_{\psi}(\boldsymbol{w}(t)) \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \,, \tag{3.12}$$

where (denoting the Jacobian of $\psi(\boldsymbol{w}(t))$ *by* $\boldsymbol{J}_{\psi}(\boldsymbol{w}(t))$*)*

$$\boldsymbol{P}_{\psi} \doteq \boldsymbol{I}_d - \boldsymbol{J}_{\psi}^{\top} \Big(\boldsymbol{J}_{\psi} \boldsymbol{H}_F^{-1} \boldsymbol{J}_{\psi}^{\top} \Big)^{-1} \boldsymbol{J}_{\psi} \boldsymbol{H}_F^{-1} \,,$$

is the projection matrix onto the tangent space at $\boldsymbol{w}(t)$*. Equivalently, the update can be written as a projected natural gradient descent update*

$$\dot{\boldsymbol{w}}(t) = -\eta \boldsymbol{P}_{\psi}(\boldsymbol{w}(t))^{\top} \boldsymbol{H}_F^{-1}(\boldsymbol{w}(t)) \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \,. \tag{3.13}$$

Proof. We use a Lagrange multiplier $\boldsymbol{\lambda}(t) \in \mathbb{R}^m$ in (3.5) to enforce the constraint $\psi(\boldsymbol{w}(t)) = \mathbf{0}$ for all $t \geq 0$,

$$\min_{\boldsymbol{w}(t)} \left\{ {}^1\!/\!_\eta \, \dot{D}_F(\boldsymbol{w}(t), \boldsymbol{w}_s) + L(\boldsymbol{w}(t)) + \boldsymbol{\lambda}(t) \cdot \psi(\boldsymbol{w}(t)) \right\}. \tag{3.14}$$

Setting the derivative w.r.t. $\boldsymbol{w}(t)$ to zero, we have

$$\dot{f}(\boldsymbol{w}(t)) + \eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) + \boldsymbol{J}_\psi^\top(\boldsymbol{w}(t)) \boldsymbol{\lambda}(t) = \mathbf{0} \,. \tag{3.15}$$

In order to solve for $\boldsymbol{\lambda}(t)$, first note that $\dot{\psi}(\boldsymbol{w}(t)) = \boldsymbol{J}_\psi(\boldsymbol{w}(t)) \, \dot{\boldsymbol{w}}(t) = \mathbf{0}$. Using the equality $\dot{f}(\boldsymbol{w}(t)) = \boldsymbol{H}_F(\boldsymbol{w}(t)) \dot{\boldsymbol{w}}(t)$ and multiplying both sides by $\boldsymbol{J}_\psi(\boldsymbol{w}(t)) \boldsymbol{H}_F^{-1}(\boldsymbol{w}(t))$ yields (ignoring t)

$$\underbrace{\boldsymbol{J}_\psi(\boldsymbol{w}) \, \dot{\boldsymbol{w}}}_{=\mathbf{0}} + \eta \, \boldsymbol{J}_\psi(\boldsymbol{w}) \boldsymbol{H}_F^{-1}(\boldsymbol{w}) \nabla L(\boldsymbol{w}) + \boldsymbol{J}_\psi(\boldsymbol{w}) \boldsymbol{H}_F^{-1}(\boldsymbol{w}) \boldsymbol{J}_\psi^\top(\boldsymbol{w}) \boldsymbol{\lambda}(t) = \mathbf{0}$$

Assuming that the matrix inverse exists, we can written

$$\boldsymbol{\lambda}(t) = -\eta \left(\boldsymbol{J}_\psi(\boldsymbol{w}) \boldsymbol{H}_F^{-1}(\boldsymbol{w}) \boldsymbol{J}_\psi^\top(\boldsymbol{w}) \right)^{-1} \boldsymbol{J}_\psi(\boldsymbol{w}) \boldsymbol{H}_F^{-1}(\boldsymbol{w}) \nabla L(\boldsymbol{w}) \,.$$

Plugging in for $\boldsymbol{\lambda}(t)$ yields (3.13). Multiplying both sides by $\boldsymbol{H}_F(\boldsymbol{w})$ and using $\dot{f}(\boldsymbol{w}) = \boldsymbol{H}_F(\boldsymbol{w}) \dot{\boldsymbol{w}}$ yields (3.12). □

In general, finding the solution $\boldsymbol{w}^\star(t)$ involves solving the PDE (3.1) and applying the boundary condition $\boldsymbol{w}(s) = \boldsymbol{w}_s$. However by integrating (3.1), a general form of the solution can be obtained as

$$f(\boldsymbol{w}^\star(t)) - f(\boldsymbol{w}_s) = -\eta \int_s^t \nabla_{\boldsymbol{w}} L(\boldsymbol{w}^\star(z)) \, \mathrm{d}\, z \,.$$

We obtain a discretized version of continuous-time MD by simply setting $t = s+1$:

$$f(\boldsymbol{w}^\star_{s+1}) - f(\boldsymbol{w}_s) = -\eta \int_s^{s+1} \nabla_{\boldsymbol{w}} L(\boldsymbol{w}^\star(z))\,\mathrm{d}\,z\,. \tag{3.16}$$

We can recover the implicit MD update (3.3) by approximating the Riemann integral using the value of the gradient at $z = s + 1$. Alternatively, the explicit MD update (3.4) can be obtained by an approximation at $z = s$. In the following, we contrast all three versions of discrete mirror descent updates for the case of linear regression with the identity link.

Example 2 (Linear Regression Using GD)**.** *We consider solving the linear regression (LR) problem using GD updates. Given a set of input-output pairs of the form* $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ *where* $\boldsymbol{x}_n \in \mathbb{R}^d$ *and* $y_n \in \mathbb{R}$*, the goal is to find the weight vector that minimizes the squared loss*

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2\,, \tag{3.17}$$

where $\boldsymbol{X}$ *contains the instance vectors as rows* $\boldsymbol{X}_{n,:} = \boldsymbol{x}_n^\top$ *and* $\boldsymbol{y} \in \mathbb{R}^N$ *denotes the vector of outputs. We make the assumption that* $N \geq d$ *and* X *is in general position. The analysis for the underdetermined case follows similarly. The least-squared solution has a closed form in terms of the Moore–Penrose inverse of* $\boldsymbol{X}$*, that is*

$$\boldsymbol{w}^\star = \boldsymbol{X}^\dagger \boldsymbol{y} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}\,.$$

The iterative explicit GD update for solving the LR problem involves linear approximation of the loss function (3.17) *at the current parameter* $\boldsymbol{w}_s$ *and minimizing*

the sum of the linear loss plus an inertia term

$$\begin{aligned}\boldsymbol{w}_{s+1} &= \arg\min_{\boldsymbol{w}} \frac{1}{\eta}||\boldsymbol{w}-\boldsymbol{w}_s||^2 + \Big((\boldsymbol{X}\boldsymbol{w}_s-\boldsymbol{y})^2 + (\boldsymbol{w}-\boldsymbol{w}_s)^\top \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_s-\boldsymbol{y})\Big) \\ &= \boldsymbol{w}_s - \eta\,\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_s-\boldsymbol{y}) \\ &\overset{\eta\to\infty}{=} -\eta \times \textit{old gradient}\,.\end{aligned} \tag{3.18}$$

Note that the explicit update cannot recover $\boldsymbol{w}^\star$ *in a single step with large* η*. However, the implicit GD update which is derived as*

$$\begin{aligned}\boldsymbol{w}_{s+1} &= \arg\min_{\boldsymbol{w}} \frac{1}{\eta}||\boldsymbol{w}-\boldsymbol{w}_s||^2 + (\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^2 \\ &= \boldsymbol{w}_s - \eta\,\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_{s+1}-\boldsymbol{y}) \\ &= (\boldsymbol{I}+\eta\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{w}_s+\eta\boldsymbol{X}^\top\boldsymbol{y}) \\ &= \boldsymbol{w}_s - \eta(\boldsymbol{I}+\eta\boldsymbol{X}^\top\boldsymbol{X})^{-1}\,\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_s-\boldsymbol{y}) \\ &\overset{\eta\to\infty}{=} (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{X}^\dagger\boldsymbol{y}\,,\end{aligned} \tag{3.19}$$

recovers the least-squared solution in a single step as $\eta\to\infty$*. In contrast, the explicit update simply follows the old gradient and requires iteration for convergence.*

A more precise approach for performing GD updates for LR involves solving the following ODE [45]:

$$\dot{\boldsymbol{w}}(t) = -\eta\,\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}(t)-\boldsymbol{y})\,,$$

i.e. continuous-time MD (3.1) *with the identity link. Combining the homogeneous*

and particular solutions

$$\begin{cases} \boldsymbol{w}^H(t) = \mathbf{exp}\Big(-\eta\, \boldsymbol{X}^\top \boldsymbol{X}\,(t-s)\Big)\,\boldsymbol{c} \quad \Big(\textit{with } \boldsymbol{c} \in \mathbb{R}^d\Big)\,, & \textit{(homogeneous)} \\ \boldsymbol{w}^P(t) = \boldsymbol{X}^\dagger \boldsymbol{y}\,, & \textit{(particular)} \end{cases}$$

and enforcing the boundary condition: $\boldsymbol{w}(s) = \boldsymbol{w}_s$ *yields*

$$\boldsymbol{w}(t) = \mathbf{exp}\Big(-\eta\, \boldsymbol{X}^\top \boldsymbol{X}\,(t-s)\Big)(\boldsymbol{w}_s - \boldsymbol{X}^\dagger \boldsymbol{y}) + \boldsymbol{X}^\dagger \boldsymbol{y}\,. \tag{3.20}$$

Note that this continuous-time GD update (3.20) *converges to the least-squared solution when* $\eta \to \infty$*. Additionally, the explicit and implicit updates can be recovered as approximations of the continuous-time update* (3.20) *as follows. For* $t = s+1$*, we obtain the explicit GD update* (3.18) *by using first order Taylor series approximation of the matrix exponential:*

$$\boldsymbol{w}_{s+1} \approx (\mathbf{I} - \eta\, \boldsymbol{X}^\top \boldsymbol{X})(\boldsymbol{w}_s - \boldsymbol{X}^\dagger \boldsymbol{y}) + \boldsymbol{X}^\dagger \boldsymbol{y} = \boldsymbol{w}_s - \eta\, \boldsymbol{X}^\top \Big(\boldsymbol{X}\boldsymbol{w}_s - \boldsymbol{y}\Big)\,.$$

Also, by first inverting the matrix exponential and then using the first order Taylor expansion, we obtain the implicit GD update (3.19)*:*

$$\begin{aligned} \boldsymbol{w}_{s+t} &= \mathbf{exp}(+\eta\, \boldsymbol{X}^\top \boldsymbol{X})^{-1}\,(\boldsymbol{w}_s - \boldsymbol{X}^\dagger \boldsymbol{y}) + \boldsymbol{X}^\dagger \boldsymbol{y} \\ &\approx (\mathbf{I} + \eta\, \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{w}_s - \boldsymbol{X}^\dagger \boldsymbol{y}) + \boldsymbol{X}^\dagger \boldsymbol{y} \\ &= (\mathbf{I} + \eta\, \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\cancel{\boldsymbol{w}_s - \boldsymbol{X}^\dagger \boldsymbol{y}} + (\cancel{\boldsymbol{I}} + \eta \boldsymbol{X}^\top \boldsymbol{X})\, \boldsymbol{X}^\dagger \boldsymbol{y}) \\ &= (\mathbf{I} + \eta\, \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{w}_s + \eta\, \boldsymbol{X}^\top \boldsymbol{y})\,. \end{aligned}$$

3.3 Discretized Updates

In this section, we discuss different ways of discretizing the CMD updates. Specifically, we compare the direct discretization of the functional form (3.5) and the corresponding Euler approximations of the corresponding CMD and its natural gradient form. We also briefly discuss the normalized updates.

The most straight-forward discretization of the unconstrained CMD update (3.1) is the forward Euler (i.e. explicit) discretization, given in (3.4). Note that this corresponds to a (approximate) minimizer of the discretized form of (3.5), that is,

$$\arg\min_{\boldsymbol{w}} \left\{ {}^{1}\!/_{\eta} \left(D_F(\boldsymbol{w}, \boldsymbol{w}_s) - \underbrace{D_F(\boldsymbol{w}_s, \boldsymbol{w}_s)}_{=0} \right) + L(\boldsymbol{w}) \right\}.$$

An alternative way of discretizing is to apply the approximation on the equivalent natural gradient form (3.9), which yields

$$\boldsymbol{w}_{s+1} - \boldsymbol{w}_s = -\eta\, \boldsymbol{H}_F^{-1}(\boldsymbol{w}_s)\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)\,.$$

Despite being equivalent in continuous-time, the two approximations may correspond to different updates after discretization. As an example, for the EGU update motivated by $f(\boldsymbol{w}) = \log \boldsymbol{w}$ link, the latter approximation yields

$$\boldsymbol{w}_{s+1} = \boldsymbol{w}_s \odot \left(\mathbf{1} - \eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s) \right),$$

which corresponds to the unnormalized *prod* update, introduced by [18] as a Taylor approximation of the original EGU update.

The situation becomes more involved for discretizing the constrained updates. As the first approach, it is possible to directly discretize the projected CMD

update (3.12)

$$f(\tilde{\boldsymbol{w}}_{s+1}) - f(\boldsymbol{w}_s) = -\eta\, \boldsymbol{P}_\psi(\boldsymbol{w}_s)\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)\,.$$

However, note that the new parameter $\tilde{\boldsymbol{w}}_{w+1}$ may fall outside the constraint set $\mathcal{D}_\psi \doteq \{\boldsymbol{w} \in \mathcal{D} | \, \psi(\boldsymbol{w}) = \boldsymbol{0}\}$. As a result, a Bregman projection [59] into $\mathcal{D}_\psi$ may need to be applied after the update, that is

$$\boldsymbol{w}_{s+1} = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathcal{D}_\psi} D_F(\boldsymbol{w}, \tilde{\boldsymbol{w}}_{s+1})\,. \tag{3.21}$$

As an example, for the normalized EG updates with the additional constraint that $\boldsymbol{w}\cdot\boldsymbol{1} = 1$, we have $\boldsymbol{P}_\psi(\boldsymbol{w}) = \boldsymbol{I}_n - \boldsymbol{1}\boldsymbol{w}^\top$ and the approximation yields

$$\log(\tilde{\boldsymbol{w}}_{s+1}) - \log(\boldsymbol{w}_s) = -\eta\left(\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s) - \boldsymbol{1}\,\mathbb{E}_{\boldsymbol{w}_s}[\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)]\right),$$

where $\mathbb{E}_{\boldsymbol{w}_s}[\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)] = \boldsymbol{w}_s \cdot \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)$. Clearly, $\tilde{\boldsymbol{w}}_{s+1}$ may not necessarily satisfy $\tilde{\boldsymbol{w}}_{s+1}\cdot\boldsymbol{1} = 1$. Therefore, we apply

$$\boldsymbol{w}_{s+1} = \frac{\tilde{\boldsymbol{w}}_{s+1}}{\|\tilde{\boldsymbol{w}}_{s+1}\|_1}\,,$$

which corresponds to the Bregman projection onto the unit simplex using the relative entropy divergence [40].

An alternative approach for discretizing the constrained update would be to first discretize the functional objective with the Lagrange multiplier (3.14) and then (approximately) solve for the update. That is,

$$\boldsymbol{w}_{s+1} = \operatorname*{arg\,min}_{\boldsymbol{w}} \left\{ {}^1\!/\!_\eta \left(D_F(\boldsymbol{w}, \boldsymbol{w}_s) - \underbrace{D_F(\boldsymbol{w}_s, \boldsymbol{w}_s)}_{=0} \right) + L(\boldsymbol{w}) + \boldsymbol{\lambda}\cdot\psi(\boldsymbol{w}) \right\}.$$

Note that in this case, the update satisfies the constraint $\psi(\boldsymbol{w}_{s+1}) = \boldsymbol{0}$ because

of directly using the Lagrange multiplier. For the normalized EG update, this corresponds to the original normalized EG update in [46],

$$\boldsymbol{w}_{s+1} = \frac{\boldsymbol{w}_s \odot \exp\Big(-\eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)\Big)}{\|\boldsymbol{w}_s \odot \exp\Big(-\eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)\Big)\|_1}\,.$$

Finally, it is also possible to discretized the projected natural gradient update (3.13). Again, a Bregman projection into $\mathcal{D}_\psi$ may be required after the update, that is,

$$\tilde{\boldsymbol{w}}_{s+1} - \boldsymbol{w}_s = -\eta \boldsymbol{P}_\psi(\boldsymbol{w}_s)^\top \boldsymbol{H}_F^{-1}(\boldsymbol{w}_s) \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))\,,$$

followed by (3.21). For the normalized EG update, the first step corresponds to

$$\boldsymbol{w}_{s+1} = \boldsymbol{w}_s \odot \Big(\mathbf{1} - \eta\Big(\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s) - \mathbf{1}\, \mathbb{E}_{\boldsymbol{w}_s}[\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_s)]\Big)\Big)\,,$$

which recovers to the *approximated EG* update of [40]. Note that $\boldsymbol{w}_{s+1} \cdot \mathbf{1} = 1$ and therefore, no projection step is required in this particular case.

Although different ways of discretizing the CMD update (3.1) are equivalent in the continuous-time limit, they yield remarkably different behavior in the discrete case. In general, the approximations that apply the link function directly and have a more implicit form for the gradient yield better results in practice. In the normalized case, the updates that use the Lagrange multiplier to ingrain the Bregman projection are preferable over those that generally fall outside of the constraint set at each step and require a projection step explicitly.

3.4 Reparameterization

We now establish the first main result of this chapter.

Theorem 4. *Let $\boldsymbol{w} = q(\boldsymbol{u}) \in \mathbb{R}^d$ where $\boldsymbol{u} \in \mathbb{R}^k$ with $k \geq d$ is a vector of parameters. The CMD update on $\boldsymbol{w}$ w.r.t. the convex function F,*

$$\dot{f}(\boldsymbol{w}(t)) = -\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \,,$$

coincides with the CMD update for parameters $\boldsymbol{u}$ using the convex function G (and link $g \doteq \nabla_{\boldsymbol{u}} G$) on the composite loss $L \circ q$,

$$\dot{g}(\boldsymbol{u}) = -\eta \, \nabla_{\boldsymbol{u}} L \circ q\Big(\boldsymbol{u}(t)\Big) \,,$$

provided that $\mathrm{range}(q) \subseteq \mathrm{dom}(F)$ *holds and we have $\boldsymbol{H}_F^{-1}(\boldsymbol{w}) = \boldsymbol{J}_q(\boldsymbol{u}) \, \boldsymbol{H}_G^{-1}(\boldsymbol{u}) \, \boldsymbol{J}_q(\boldsymbol{u})^\top$ for all $\boldsymbol{w} = q(\boldsymbol{u})$.*

Proof. Note that $\dot{\boldsymbol{w}}(t) = \frac{\partial \boldsymbol{w}(t)}{\partial \boldsymbol{u}(t)} \, \dot{\boldsymbol{u}}(t) = \boldsymbol{J}_q(\boldsymbol{u}(t)) \, \dot{\boldsymbol{u}}(t)$. Also, $\nabla_{\boldsymbol{u}} L \circ q\Big(\boldsymbol{u}(t)\Big) = \boldsymbol{J}_q(\boldsymbol{u}(t))^\top \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))$. The CMD update on $\boldsymbol{u}$ with the link function $g(\boldsymbol{u})$ can be written as $\dot{\boldsymbol{u}}(t) = -\eta \, \boldsymbol{H}_G^{-1}(\boldsymbol{u}(t)) \nabla_{\boldsymbol{u}} L \circ q\Big(\boldsymbol{u}(t)\Big)$. Thus (dropping t for simplicity),

$$\dot{\boldsymbol{u}} = -\eta \, \boldsymbol{H}_G^{-1}(\boldsymbol{u}) \, \boldsymbol{J}_q(\boldsymbol{u})^\top \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \,.$$

Multiplying by $\boldsymbol{J}_q(\boldsymbol{u})$ from the left yields

$$\dot{\boldsymbol{w}} = -\eta \, \boldsymbol{J}_q(\boldsymbol{u}) \boldsymbol{H}_G^{-1}(\boldsymbol{u}) \boldsymbol{J}_q(\boldsymbol{u})^\top \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \,.$$

Comparing the result to (3.9) concludes the proof. □

In the following, we provide a number of examples.

Example 3 (EGU as GD)**.** *The continuous-time EGU can be reparameterized as continuous-time GD with the reparameterization function $\boldsymbol{w} = q(\boldsymbol{u}) = {}^1\!/_4 \, \boldsymbol{u} \odot \boldsymbol{u} =$*

$^1/_4\,\boldsymbol{u}^{\odot 2}$, *i.e.*

$$\dot{\log}(\boldsymbol{w}) = -\eta\,\nabla L(\boldsymbol{w}) \;\; \textit{equals} \;\; \dot{\boldsymbol{u}} = -\eta\,\underbrace{\nabla L \circ q\,(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}} L\,(^1/_4\,\boldsymbol{u}^{\odot 2})} = -^\eta/_2\,\boldsymbol{u} \odot \nabla L(\boldsymbol{w})$$

This is proven by verifying the condition of Theorem 4:

$$\boldsymbol{J}_q(\boldsymbol{u})\boldsymbol{J}_q(\boldsymbol{u})^\top = {}^1/_2\,\mathrm{diag}(\boldsymbol{u})\,({}^1/_2\,\mathrm{diag}(\boldsymbol{u}))^\top = \mathrm{diag}({}^1/_4\,\boldsymbol{u}^{\odot 2}) = \mathrm{diag}(\boldsymbol{w}) = \boldsymbol{H}_F^{-1}(\boldsymbol{w})\,.$$

Example 4 (Reduced EG in 2-dimension)**.** *Consider the* 2*-dimensional normalized weights* $\boldsymbol{w} = [\,\omega, 1-\omega]^\top$ *where* $0 \le \omega \le 1$*. The normalized reduced EG update [69] is motivated by the link function* $f(w) = \log \frac{w}{1-w}$*, thus* $H_F(w) = \frac{1}{w} + \frac{1}{1-w} = \frac{1}{w(1-w)}$*. This update can be reparameterized as a GD update on* $u \in \mathbb{R}$ *via* $\omega = q(u) = {}^1/_2\,(1+\sin(u))$ *i.e.*

$$\dot{\log}(\frac{w}{1-w}) = -\eta\,\nabla_w L(w) \;\; \textit{equals} \;\; \dot{u} = -\eta\,\underbrace{\nabla_u L \circ q\,(u)}_{\nabla_u L\,\left({}^1/_2\,(1+\sin(u))\right)} = -\eta\,\frac{\cos(u)}{2}\nabla L(w)\,.$$

This is verified by checking the condition of Theorem 4: $J_q(u) = {}^1/_2\,\cos(u)$ *and*

$$J_q(u)J_q(u)^\top = \frac{1}{4}\,\cos^2(u) = \frac{1}{2}\,\Big(1+\sin(u)\Big)\,\frac{1}{2}\Big(1-\sin(u)\Big) = w(1-w) = H_F^{-1}(w)\,.$$

Open problem The generalization of the reduced EG link function to $d > 2$ dimensions becomes $f(\boldsymbol{w}) = \log \frac{\boldsymbol{w}}{1-\sum_{i=1}^{d-1} w_i}$ which utilizes the first $(d-1)$-dimensions $\boldsymbol{w}$ s.t. $[\boldsymbol{w}^\top, w_d]^\top \in \Delta^{d-1}$. Reparameterizing the CMD update using

this link as CGD is open. The update can be reformulated as

$$\begin{aligned}\dot{\boldsymbol{w}} &= -\eta \left(\operatorname{diag}\Big(\frac{1}{\boldsymbol{w}}\Big) + \frac{1}{1-\sum_{i=1}^{d-1} w_i}\, \mathbf{1}\mathbf{1}^\top\right)^{-1} \nabla L(\boldsymbol{w}) \\ &= -\eta \left(\operatorname{diag}(\boldsymbol{w}) - \boldsymbol{w}\boldsymbol{w}^\top\right) \nabla\, L(\boldsymbol{w})\,.\end{aligned}$$

Later, we will give an d-dimensional version of EG using a projection onto a constraint.

Example 5 (Burg updates as GD)**.** *The update associated with the negative Burg entropy $F(\boldsymbol{w}) = -\sum_{i=1}^{d} \log w_i$ and link $f(\boldsymbol{w}) = -\mathbf{1} \oslash \boldsymbol{w}$ is reparameterized as GD with $\boldsymbol{w} = q(\boldsymbol{u}) := \exp(\boldsymbol{u})$, i.e.*

$$(-\mathbf{1} \oslash \boldsymbol{w})\dot{} = -\eta\, \nabla L(\boldsymbol{w}) \text{ equals } \dot{\boldsymbol{u}} = -\eta\, \underbrace{\nabla L \circ q\,(\boldsymbol{u})}_{\nabla_{\boldsymbol{u}} L\,(\exp(\boldsymbol{u}))} = -\eta\, \exp(\boldsymbol{u}) \odot \nabla L(\boldsymbol{w})\,,$$

This is verified by the condition of Theorem 4: $\boldsymbol{H}_F(\boldsymbol{w}) = \operatorname{diag}(\mathbf{1} \oslash \boldsymbol{w})^2$, $\boldsymbol{J}_q(\boldsymbol{u}) = \operatorname{diag}(\exp(\boldsymbol{u}))$, and

$$\boldsymbol{J}_q(\boldsymbol{u})\boldsymbol{J}_q(\boldsymbol{u})^\top = \operatorname{diag}(\exp(\boldsymbol{u}))^2 = \operatorname{diag}(\boldsymbol{w})^2 = \boldsymbol{H}_F^{-1}(\boldsymbol{w})\,.$$

Example 6 (EGU as Burg)**.** *The reparameterization step can be chained, and applied in reverse, when the reparameterization function q is invertible. For instance, we can first apply the inverse reparameterization of the Burg update as GD from Example 5, i.e. $\boldsymbol{u} = q^{-1}(\boldsymbol{w}) = \log \boldsymbol{w}$. Subsequently, applying the reparameterization of EGU as GD from Example 3, i.e. $\boldsymbol{v} = \tilde{q}(\boldsymbol{u}) = {}^1\!/_4\, \boldsymbol{u}^{\odot 2}$, results*

in the reparameterization of EGU as Burg update, that is,

$$\dot{\log}(\boldsymbol{v}) = -\eta\, \nabla L(\boldsymbol{v}) \quad \textit{equals} \left(-\dot{\frac{1}{\boldsymbol{w}}}\right) = -\eta\, \underbrace{\nabla_{\boldsymbol{w}} L \circ \tilde{q} \circ q^{-1}(\boldsymbol{w})}_{\nabla_{\boldsymbol{w}} L(1/4(\log \boldsymbol{w})^{\odot 2})} \\ = -\eta\, (\log(\boldsymbol{w}) \oslash (2\boldsymbol{w})) \odot \nabla L(\boldsymbol{v})\,.$$

Note that Theorem 4 shows, in general, how the local geometry is affected by the reparameterization function q. For instance, the primal (3.1) and dual (3.11) equivalence is an immediate consequence [4]:

Corollary 6. *The primal* (3.1) *and dual* (3.11) *CMD updates are equivalent via the special case of reparameterization where* $\boldsymbol{w}(t) = f^*\Big(\boldsymbol{\theta}(t)\Big)$ *(or* $\boldsymbol{\theta}(t) = f(\boldsymbol{w}(t))$*).*

In the following, we also provide the constrained reparameterized updates for completeness.

Theorem 5. *The constrained CMD update* (3.12) *coincides with the reparameterized projected gradient update on the composite loss,*

$$\dot{g}\Big(\boldsymbol{u}(t)\Big) = -\eta\, \boldsymbol{P}_{\psi \circ q}(\boldsymbol{u}(t)) \nabla_{\boldsymbol{u}} L \circ q(\boldsymbol{u}(t))\,,$$

where $(\boldsymbol{J}_{\psi \circ q}(\boldsymbol{u}(t)) \doteq \boldsymbol{J}_q^\top(\boldsymbol{u}(t)) \nabla_{\boldsymbol{w}} \psi\Big(q(\boldsymbol{u}(t))\Big))$

$$\boldsymbol{P}_{\psi \circ q} \doteq \boldsymbol{I}_k - \boldsymbol{J}_{\psi \circ q}^\top \Big(\boldsymbol{J}_{\psi \circ q} \boldsymbol{H}_G^{-1} \boldsymbol{J}_{\psi \circ q}^\top\Big)^{-1} \boldsymbol{J}_{\psi \circ q} \boldsymbol{H}_G^{-1}\,,$$

is the projection matrix onto the tangent space at $\boldsymbol{u}(t)$*.*

Proof. Similar to the proof of Proposition 4, we use a Lagrange multiplier $\boldsymbol{\lambda}(t) \in$

[4] Note that the equivalence of the primal-dual updates was already shown in [69] for the continuous case and in [54] for the discrete case (where it is only one-sided).

$\mathbb{R}^m$ to enforce the constraint $\psi \circ q(\boldsymbol{u}(t)) = \mathbf{0}$ for all $t \geq 0$,

$$\min_{\boldsymbol{u}(t)} \left\{ {}^1\!/\!_\eta \, \dot{D}_G(\boldsymbol{u}(t), \boldsymbol{u}_s) + L \circ q(\boldsymbol{u}(t)) + \boldsymbol{\lambda}(t) \cdot \psi \circ q(\boldsymbol{u}(t)) \right\}.$$

Setting the derivative w.r.t. $\boldsymbol{u}(t)$ to zero, we have

$$\dot{g}(\boldsymbol{w}(t)) + \eta \, \nabla_{\boldsymbol{u}} L \circ q(\boldsymbol{w}(t)) + \boldsymbol{J}_{\psi \circ q}^\top(\boldsymbol{u}(t)) \boldsymbol{\lambda}(t) = \mathbf{0}\,,$$

where $\boldsymbol{J}_{\psi \circ q}(\boldsymbol{u}) \doteq \boldsymbol{J}_q^\top(\boldsymbol{u}) \boldsymbol{J}_\psi(q(\boldsymbol{u}))$. In order to solve for $\boldsymbol{\lambda}(t)$, we use the fact that $\dot{\psi \circ q}(\boldsymbol{u}(t)) = \boldsymbol{J}_{\psi \circ q}(\boldsymbol{u}(t)) \, \dot{\boldsymbol{u}}(t) = \mathbf{0}$. Using the equality $\dot{g}(\boldsymbol{u}(t)) = \boldsymbol{H}_G(\boldsymbol{u}(t)) \dot{\boldsymbol{u}}(t)$ and multiplying both sides by $\boldsymbol{J}_{\psi \circ q}(\boldsymbol{u}(t)) \boldsymbol{H}_G^{-1}(\boldsymbol{u}(t))$ yields (ignoring t),

$$\boldsymbol{J}_{\psi \circ q}(\boldsymbol{u}) \, \dot{\boldsymbol{u}} + \eta \, \boldsymbol{J}_{\psi \circ q}(\boldsymbol{w}) \boldsymbol{H}_G^{-1}(\boldsymbol{u}) \nabla L \circ q(\boldsymbol{u}) + \boldsymbol{J}_{\psi \circ q}(\boldsymbol{w}) \boldsymbol{H}_G^{-1}(\boldsymbol{w}) \boldsymbol{J}_{\psi \circ q}^\top(\boldsymbol{u}) \boldsymbol{\lambda}(t) = \mathbf{0}\,.$$

The rest of the proof follows similarly by solving for $\boldsymbol{\lambda}(t)$ and rearranging the terms. Applying the results of Theorem 4 concludes the proof. □

Example 7 (EG as GD)**.** *The normalized EG update is motivated similar to the EGU update with the additional constraint that* $\boldsymbol{w} \cdot \mathbf{1}_d = 1$*. Note that for the negative entropy function,* $\boldsymbol{H}_F = \operatorname{diag}(\boldsymbol{w})^{-1}$*. Using Proposition 4, we have*

$$\boldsymbol{P}_\psi = \boldsymbol{I}_d - \frac{\mathbf{1}\mathbf{1}^\top \operatorname{diag}(\boldsymbol{w})}{\mathbf{1}^\top \operatorname{diag}(\boldsymbol{w}) \mathbf{1}} = \boldsymbol{I}_d - \mathbf{1}\boldsymbol{w}^\top\,.$$

Thus, the projected update can be written as,

$$\begin{aligned} \dot{\boldsymbol{w}}(t) &= -\eta \, \boldsymbol{P}_\psi^\top \operatorname{diag}(\boldsymbol{w}) \nabla L(\boldsymbol{w}) \\ &= -\eta \left(\boldsymbol{I}_d - \boldsymbol{w}\mathbf{1}^\top \right) \operatorname{diag}(\boldsymbol{w}) \nabla L(\boldsymbol{w}) \\ &= -\eta \left(\operatorname{diag}(\boldsymbol{w}) - \boldsymbol{w}\boldsymbol{w}^\top \right) \nabla L(\boldsymbol{w})\,. \end{aligned} \tag{3.22}$$

Note that the projection ensures that $\dot{\boldsymbol{w}}(t) \cdot \mathbf{1} = 0$ *and therefore, a change in* $\boldsymbol{w}(t)$ *with an infinitesimal step size,* $\boldsymbol{w}(t) + \mathrm{d}\boldsymbol{w}(t)$*, remains on the simplex.*

We now develop the projected reparameterized updates for EG using Theorem 5 and show the equivalence to update (3.22)*. Let* $\boldsymbol{w} = q(\boldsymbol{u}) = {}^1\!/_4\, \boldsymbol{u} \odot \boldsymbol{u}$ *s.t.* $\boldsymbol{u} \in \mathbb{R}^d$ *and* $\|\boldsymbol{u}\|_2 = 2$*. Clearly, this choice for* $\boldsymbol{u}$ *results in* $\boldsymbol{w} \in \Delta^{d-1}$*. Note that for GD updates on* $\boldsymbol{u}$*, we have* $\boldsymbol{H}_G = \boldsymbol{I}_d$*. Using Theorem 5, we have*

$$\boldsymbol{J}_{\psi \circ q} = \frac{1}{2}\, \boldsymbol{u}^\top \quad \textit{and} \quad \boldsymbol{P}_{\psi \circ q} = \boldsymbol{I}_d - \frac{1}{4}\, \boldsymbol{u}\boldsymbol{u}^\top .$$

This results in the following projected updates for $\boldsymbol{u}$

$$\dot{\boldsymbol{u}}(t) = -\eta \left(\boldsymbol{I}_d - \frac{1}{4}\, \boldsymbol{u}\boldsymbol{u}^\top \right) \nabla_{\boldsymbol{u}} L \circ q(\boldsymbol{u}) \,. \tag{3.23}$$

Using the fact that $\dot{\boldsymbol{w}} = {}^1\!/_2\, \boldsymbol{u} \odot \dot{\boldsymbol{u}}$ *and* $\nabla_{\boldsymbol{u}} L \circ q(\boldsymbol{u}) = {}^1\!/_2\, \boldsymbol{u} \odot \nabla_{\boldsymbol{w}} L(\boldsymbol{w})$*, we have,*

$$\begin{aligned} \dot{\boldsymbol{w}}(t) &= -\eta\, \frac{1}{4}\, \mathrm{diag}(\boldsymbol{u}) \Big(\boldsymbol{I}_d - \frac{1}{4}\, \boldsymbol{u}\boldsymbol{u}^\top \Big)\, \mathrm{diag}(\boldsymbol{u}) \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \\ &= -\eta \left(\mathrm{diag}(\boldsymbol{w}) - \boldsymbol{w}\boldsymbol{w}^\top \right) \nabla L(\boldsymbol{w}) \,, \end{aligned}$$

which is the same as update (3.22)*.*

So far, we discussed reparameterizing a number of important CMD updates, such as EGU and the update motivated using the Burg divergence, as GD. We now show that EGU and many other updates can be unified as GD via a simple reparameterization. For this, we first introduce the tempered Bregman divergence.

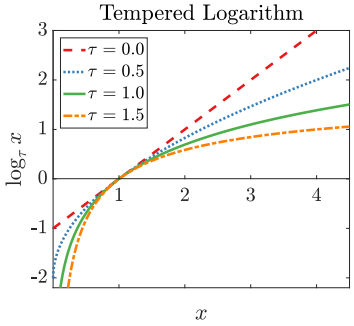


Figure 3.2: $\log_\tau(x)$, for different $\tau \geq 0$.

3.5 Tempered Bregman Updates

In this section, we introduce a tempered Bregman divergence that includes the squared Euclidean and the relative entropy divergences as special cases. As a result, the new Bregman divergence allows us to derive updates that interpolate between GD and EGU. Next, we show the reparameterization of the tempered updates and unify all the updates as GD updates on the reparameterized weights.

The tempered relative entropy divergence [5] is defined based on the tempered logarithm link function [49] which generalizes the standard log function:

$$f_\tau(\boldsymbol{w}) = \log_\tau(\boldsymbol{w}) = \frac{1}{1-\tau}(\boldsymbol{w}^{1-\tau} - 1)\,, \tag{3.24}$$

for $\boldsymbol{w} \in \mathbb{R}^d_{\geq 0}$ and $\tau \in \mathbb{R}$. The $\log_\tau$ function is shown in Figure 3.2 for different values of $\tau \geq 0$. Note that $\tau = 1$ recovers the standard log function as a limit point. The $\log_\tau(\boldsymbol{w})$ link function is the gradient of the convex function

$$\begin{aligned} F_\tau(\boldsymbol{w}) &= \sum_i \Big(w_i \log_\tau w_i + \frac{1}{2-\tau}\,(1 - w_i^{2-\tau}) \Big) \\ &= \sum_i \Big(\frac{1}{(1-\tau)(2-\tau)}\, w_i^{2-\tau} - \frac{1}{1-\tau}\, w_i + \frac{1}{2-\tau} \Big)\,. \end{aligned}$$

τ	$F_\tau(\boldsymbol{w})$	$D_{F_\tau}(\tilde{\boldsymbol{w}}, \boldsymbol{w})$	Name
-1	$\sum_i \left(\frac{1}{6}w_i^3 - \frac{1}{2}w_i\right)$	$\sum_i \left(\frac{1}{6}(\tilde{w}_i + 2w_i)(\tilde{w}_i - w_i)^2\right)$	
0	$\frac{1}{2}\sum_i w_i^2$	$\frac{1}{2}\sum_i(\tilde{w}_i - w_i)^2$	Euclidean
$\frac{1}{2}$	$\frac{1}{3}\sum_i(4\, w_i^{\frac{4}{3}} - 6\, w_i + 2)$	$\sum_i(\frac{4}{3}\tilde{w}_i^{\frac{3}{2}} - 2\tilde{w}_i\sqrt{w_i} + \frac{3}{2}w_i^{\frac{3}{2}})$	
1	$\sum_i(w_i \log w_i - w_i + 1)$	$\sum_i(\tilde{w}_i \log \frac{\tilde{w}_i}{w_i} - \tilde{w}_i + w_i)$	KL-divergence
$\frac{3}{2}$	$\sum_i(-4\, w_i^{\frac{3}{2}} + 2\, w_i + 2)$	$2\sum_i \frac{(\sqrt{\tilde{w}_i} - \sqrt{w_i})^2}{\sqrt{w_i}}$	Squared Xi on roots
2	$\sum_i(-\log w_i + w_i)$	$\sum_i(\frac{\tilde{w}_i}{w_i} - \log \frac{\tilde{w}_i}{w_i} - 1)$	Itakura-Saito
3	$\frac{1}{2}\sum_i(-\frac{1}{w_i} + w_i - 2)$	$\frac{1}{2}\sum_i(\frac{1}{\tilde{w}_i} - \frac{2}{w_i} + \frac{\tilde{w}_i}{w_i^2})$	Inverse

Table 3.1: Some special cases of the tempered Bregman divergence.

The convex function F_τ induces the following tempered Bregman divergence[5]:

$$\begin{aligned} D_{F_\tau}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) =& \sum_i \left(\tilde{w}_i \log_\tau \tilde{w}_i - \tilde{w}_i \log_\tau w_i - \frac{\tilde{w}_i^{2-\tau} - w_i^{2-\tau}}{2-\tau} \right) \\ =& \frac{1}{1-\tau} \sum_i \left(\frac{\tilde{w}_i^{2-\tau} - w_i^{2-\tau}}{2-\tau} - (\tilde{w}_i - w_i)\, w_i^{1-\tau} \right). \end{aligned} \tag{3.25}$$

For $\tau = 0$, we obtain the squared Euclidean divergence $D_{F_0}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = \frac{1}{2}\sum_i(\tilde{w}_i - w_i)^2$ and for $\tau = 1$, the relative entropy $D_{F_1}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = \sum_i(\tilde{w}_i \log(\tilde{w}_i/w_i) - \tilde{w}_i + w_i)$. Some special cases of the tempered divergence are shown in Table 3.1 for different values of τ.

In the following, we derive the CMD updates using the time derivative of (3.25) as the tempered Bregman momentum. Notice that the link function $\log_\tau(x)$ is only defined for $x \geq 0$ when $\tau > 0$. In order to have a weight $\boldsymbol{w} \in \mathbb{R}^d$, we use the EGU$^\pm$ trick [40] by maintaining two non-negative weights $\boldsymbol{w}_+$ and $\boldsymbol{w}_-$ and setting $\boldsymbol{w} = \boldsymbol{w}_+ - \boldsymbol{w}_-$. We call this the *tempered EGU$^\pm$* update. As our second main result, we show that that continuous-time tempered EGU$^\pm$ updates interpolate between continuous-time GD and continuous-time EGU (for $\tau \in [0, 1]$). Furthermore, these

[5]The second form is more commonly known as β-divergence [20] with $\beta = 2 - \tau$.

updates can be simulated by continuous-time GD on a new set of parameters $\boldsymbol{u}$ using a simple reparameterization. We show that on the underdetermined linear regression problem, under certain assumptions, these updates converge to the solution with the smallest $\mathrm{L}_{2-\tau}$-norm.

3.5.1 Tempered EGU$^{\pm}$

We first introduce the generalization of the EGU$^{\pm}$ updates using the tempered Bregman divergence (3.25). Let $\boldsymbol{w}(t) = \boldsymbol{w}_+(t) - \boldsymbol{w}_-(t)$ with $\boldsymbol{w}_+(t), \boldsymbol{w}_-(t) \in \mathbb{R}^d_{\geq 0}$ and $\boldsymbol{w}_+(0) = \boldsymbol{w}_-(0) = \boldsymbol{w}_0$. The tempered EGU$^{\pm}$ updates are motivated by

$$\operatorname*{arg\,min}_{\boldsymbol{w}_+, \boldsymbol{w}_- \in \mathbb{R}^d_{\geq 0}} \left\{ 1/\eta \, \dot{D}_{F_\tau}\Big(\boldsymbol{w}_+(t), \boldsymbol{w}_0\Big) + 1/\eta \, \dot{D}_{F_\tau}\Big(\boldsymbol{w}_-(t), \boldsymbol{w}_0\Big) + L(\boldsymbol{w}(t)) \right\}.$$

Enforcing the constraints using the Lagrange multipliers $\boldsymbol{\lambda}_+(t), \boldsymbol{\lambda}_-(t) \in \mathbb{R}^d_{\geq 0}$, we have

$$\begin{aligned} \dot{\log_\tau} \boldsymbol{w}_+(t) &= -\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) + \boldsymbol{\lambda}_+(t)\,, \\ \dot{\log_\tau} \boldsymbol{w}_-(t) &= +\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) + \boldsymbol{\lambda}_-(t)\,. \end{aligned} \tag{3.26}$$

Using $\dot{\log_\tau} \boldsymbol{w}(t) = \dot{\boldsymbol{w}}(t) \oslash \boldsymbol{w}(t)^{\odot \tau}$ and applying the KKT conditions $\boldsymbol{w}_+(t) \odot \boldsymbol{\lambda}_+(t) = \mathbf{0}$ and $\boldsymbol{w}_-(t) \odot \boldsymbol{\lambda}_-(t) = \mathbf{0}$ gives

$$\begin{aligned} \dot{\boldsymbol{w}}_+(t) &= (\boldsymbol{w}_+(t))^{\odot \tau} \odot \Big(-\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \Big)\,, \\ \dot{\boldsymbol{w}}_-(t) &= (\boldsymbol{w}_-(t))^{\odot \tau} \odot \Big(+\eta \, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)) \Big)\,, \end{aligned} \tag{3.27}$$

which means we can simply ignore $\boldsymbol{\lambda}_+(t)$ and $\boldsymbol{\lambda}_-(t)$ in (3.26). Integrating (3.26) over t and applying the inverse link yields

$$\begin{aligned} \boldsymbol{w}_+(t) &= \exp_\tau\Big(\log_\tau \boldsymbol{w}_0 - \eta\int_0^t \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(z))\,\mathrm{d}z\Big), \\ \boldsymbol{w}_-(t) &= \exp_\tau\Big(\log_\tau \boldsymbol{w}_0 + \eta\int_0^t \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(z))\,\mathrm{d}z\Big), \end{aligned} \tag{3.28}$$

where $\exp_\tau(x) \doteq [1 + (1-\tau)x]_+^{\frac{1}{1-\tau}}$. Note that $\tau = 1$ is a limit case which recovers the standard exp function and the updates (3.28) become the standard EGU$^\pm$. Additionally, the GD updates are recovered at $\tau = 0$. As a result, the tempered EGU$^\pm$ updates (3.28) interpolate between GD and EGU$^\pm$ for $\tau \in [0, 1]$ and generalize beyond for values of $\tau > 1$ and $\tau < 0$.

Note that the tempered EGU$^\pm$ updates make use of two d-dimensional non-negative weight vectors and then use a d-dimensional difference for inference. The EGU$^\pm$ updates (for which the double weight trick was introduced originally) can be reformulated using the d-dimensional link function $\operatorname{arcsinh}(\boldsymbol{w})$ which behaves as a "two-sided logarithm". Also the associated $\operatorname{arcsinh}(\boldsymbol{w})$ based Bregman divergence [71, 30] can be used to analyze EGU$^\pm$. The two-sided generalization of the tempered logarithm function is an interesting open problem.

3.5.2 Reparameterized Tempered EGU$^\pm$

We now show that the updates (3.26) can be obtained via a reparameterization on a set of weights $\boldsymbol{u}_+, \boldsymbol{u}_- \in \mathbb{R}^d$.

Proposition 5. *The updates* (3.26) *can be realized by GD updates on* $\boldsymbol{u}_+, \boldsymbol{u}_- \in \mathbb{R}^d$ *and setting* $\boldsymbol{w}_+ = q_\tau(\boldsymbol{u}_+)$ *and* $\boldsymbol{w}_- = q_\tau(\boldsymbol{u}_-)$ *where*

$$q_\tau(\boldsymbol{u}) = \Big(\frac{2}{2-\tau}\Big)^{-\frac{2}{2-\tau}} |\boldsymbol{u}|^{\frac{2}{2-\tau}}\,,\ \tau \neq 2\,. \tag{3.29}$$

Proof. Note that the Jacobian is

$$\boldsymbol{J}_{q_\tau}(\boldsymbol{u}_+) = \Big(\frac{2}{2-\tau}\Big)^{\frac{\tau}{2-\tau}} \operatorname{diag}\Big(\operatorname{sign}(\boldsymbol{u}_+) \odot |\boldsymbol{u}_+|^{\frac{\tau}{2-\tau}}\Big).$$

Thus, $\boldsymbol{J}_{q_\tau}(\boldsymbol{u}_+)\boldsymbol{J}_{q_\tau}^\top(\boldsymbol{u}_+) = \operatorname{diag}\Big((\boldsymbol{w}_+)^{\odot\tau}\Big)$. A similar construction holds for $\boldsymbol{w}_-$ and $\boldsymbol{u}_-$. Applying the results of Theorem 4 yields the same dynamics as in (3.26). □

For completeness, we write the reparameterized tempered EGU$^\pm$ updates as

$$\begin{aligned} \dot{\boldsymbol{u}}_+(t) &= -\eta\, \nabla_{\boldsymbol{u}_+} L\Big(q_\tau(\boldsymbol{u}_+) - q_\tau(\boldsymbol{u}_-)\Big), \\ \dot{\boldsymbol{u}}_-(t) &= -\eta\, \nabla_{\boldsymbol{u}_-} L\Big(q_\tau(\boldsymbol{u}_+) - q_\tau(\boldsymbol{u}_-)\Big). \end{aligned} \tag{3.30}$$

Note that in the reparameterized update, the non-negativity of the reparameterized weights is naturally imposed by the absolute value in the reparameterization.

Special cases of the reparameterized tempered EGU$^\pm$ updates include GD with $\tau = 0$ and EGU$^\pm$ with $\tau = 1$, where the latter corresponds to $\boldsymbol{w} = \boldsymbol{u}_+ \odot \boldsymbol{u}_+ - \boldsymbol{u}_- \odot \boldsymbol{u}_-$. This form of updates was recently discussed in [64] for sparse signal recovery from an underdetermined system of linear measurements. Similar approaches have been applied for mapping the replicator dynamics from the simplex to the unit sphere [2, 57]. Additionally, values of $0 < \tau < 1$ interpolate between GD and EGU. However, any other value of $\tau \in \mathbb{R} - \{2\}$ also corresponds to a valid update. Note that $\tau = 2$ is a special case, which according to [5] corresponds to the Burg divergence

$$D_{\text{Burg}}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = \sum_i \Big(\frac{\tilde{w}_i}{w_i} - \log\Big(\frac{\tilde{w}_i}{w_i}\Big) - 1\Big).$$

To find the update for Burg, we point out that the shifted version of reparameterization (3.29),

$$\widetilde{q}_\tau(\boldsymbol{u}) = \Big|\mathbf{1}_d + \big(\frac{2}{2-\tau}\big)^{-\frac{2}{2-\tau}} \boldsymbol{u}\Big|^{\frac{2}{2-\tau}},$$

also satisfies the conditions in Proposition 5. For this choice of reparameterization, we have $\widetilde{q}_2(\boldsymbol{u}) = \exp(\boldsymbol{u})$. For simplicity, we consider the reparameterization in Eq. (3.29) and adopt $q_2(\boldsymbol{u}) \doteq \exp(\boldsymbol{u})$ for the special case of $\tau = 2$ (as discussed in Example 5).

3.6 Minimum-norm Solutions

Gunasekar et. al [31] showed that on the underdetermined linear regression problem, using the factorization $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{U}^\top$ for the weight matrix $\boldsymbol{W}$ and applying continuous-time GD on $\boldsymbol{U}$ achieves the minimum L_1-norm solution. Note that in the vector (i.e. diagonal weight matrix) case, this corresponds to setting $\boldsymbol{w} = \boldsymbol{u} \odot \boldsymbol{u}$ and running continuous-time GD on $\boldsymbol{u}$, which according to Theorem 4 is equivalent to running EGU update on $\boldsymbol{w}$. The fact that the EGU update favors the minimum L_1-norm solution is linked to the strong-convexity of the negative entropy function $F_1(\boldsymbol{w}) = \sum_i (w_i \log w_i - w_i)$ (which induces the EGU updates) w.r.t. the L_1-norm. The focus here is to show that, under similar assumptions, the tempered updates converge to the solution with the smallest $\mathrm{L}_{2-\tau}$-norm when $\tau \in [0, 1]$. For a start, the following result establishes strong-convexity of F_τ function.

Lemma 7. *The function F_τ, with $0 \leq \tau \leq 1$, is $B^{-\tau}$–strongly convex over the set $\{\boldsymbol{w} \in \mathbb{R}^k_+ : \|\boldsymbol{w}\|_{2-\tau} \leq B\}$ w.r.t. the $\mathrm{L}_{2-\tau}$-norm.*

Proof. We have $\boldsymbol{H}_{F_\tau}(\boldsymbol{w}) = \nabla^2 F(\boldsymbol{w}) = \mathrm{diag}(\boldsymbol{w}^{-\odot\tau})$ (which validates $\boldsymbol{H}_{F_\tau}(\boldsymbol{w}) \succcurlyeq \boldsymbol{0}$). Applying Lemma 4, note that the function

$$(\nabla^2 F_\tau(\boldsymbol{w}) \cdot \tilde{\boldsymbol{w}}) \cdot \tilde{\boldsymbol{w}} = \sum_i \frac{\tilde{w}_i^2}{w_i^\tau},$$

is unbounded over the set $\mathcal{S} = \{\tilde{\boldsymbol{w}} \in \mathbb{R}^d_+ : \|\tilde{\boldsymbol{w}}\|_{2-\tau} \leq B\}$ and the minimum

happens at the boundary $\{\|\tilde{\boldsymbol{w}}\|_{2-\tau} = B\}$.

$$\min_{\tilde{\boldsymbol{w}}} \sum_i \frac{\tilde{w}_i^2}{w_i^\tau} + \gamma \left(\sum_i \tilde{w}_i^{2-\tau} - 1\right) \Rightarrow \tilde{\boldsymbol{w}} = B \, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_{2-\tau}} \,,$$

where γ is the Lagrange multiplier. Plugging in the solution yields $\sum_i \frac{\tilde{w}_i^2}{w_i^\tau} \geq \frac{1}{B^\tau} \|\tilde{\boldsymbol{w}}\|_{2-\tau}^2$. □

The strong convexity of the F_τ function w.r.t. the $\mathrm{L}_{2-\tau}$-norm suggests that the updates motivated by the tempered Bregman divergence (3.25) yield the minimum $\mathrm{L}_{2-\tau}$-norm solution in certain settings. We verify this by considering the vector and matrix updates for the linear regression problem.

3.6.1 Vector Case

Let $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ denote the set of input-output pairs and let $\boldsymbol{X}$ be the design matrix for which the n-th row is equal to $\boldsymbol{x}_n^\top$. Also, let $\boldsymbol{y}$ denote the vector of targets. Consider the tempered $\mathrm{EGU}^\pm$ updates (3.26) on the weights $\boldsymbol{w}(t) = \boldsymbol{w}_+(t) - \boldsymbol{w}_-(t)$ where $\boldsymbol{w}_+(t), \boldsymbol{w}_-(t) \geq \boldsymbol{0}$ and $\boldsymbol{w}_+(0) = \boldsymbol{w}_-(0) = \boldsymbol{w}_0$. Following (3.28), we have

$$\begin{aligned} \boldsymbol{w}_+(t) &= \exp_\tau \Big(\log_\tau \boldsymbol{w}_0 - \eta \int_0^t \boldsymbol{X}^\top \boldsymbol{\delta}(z) \, \mathrm{d}z \Big) \,, \\ \boldsymbol{w}_-(t) &= \exp_\tau \Big(\log_\tau \boldsymbol{w}_0 + \eta \int_0^t \boldsymbol{X}^\top \boldsymbol{\delta}(z) \, \mathrm{d}z \Big) \,, \end{aligned} \tag{3.31}$$

where $\boldsymbol{\delta}(t) = \boldsymbol{X}\Big(\boldsymbol{w}_+(t) - \boldsymbol{w}_-(t)\Big)$.

Theorem 6. *Consider the underdetermined linear regression problem where $N < d$. Let $\mathcal{E} = \{\boldsymbol{w} \in \mathbb{R}^d | \, \boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}\}$ be the set of solutions with zero error. Given $\boldsymbol{w}(\infty) \in \mathcal{E}$, then the tempered EGU$^\pm$ updates (3.28) with temperature $0 \leq \tau \leq 1$ and initial solution $\boldsymbol{w}_0 = \alpha \boldsymbol{1} \geq 0$ converge to the minimum $\mathrm{L}_{2-\tau}$-norm solution*

in $\mathcal{E}$ in the limit $\alpha \to 0$.

Proof. We show that the solution of the tempered EGU$^{\pm}$ satisfies the dual feasibility and complementary slackness KKT conditions for the following optimization problem (omitting t for simplicity):

$$\begin{aligned} &\min_{\boldsymbol{w}_+, \boldsymbol{w}_-} \ \|\boldsymbol{w}_+ - \boldsymbol{w}_-\|_{2-\tau}^{2-\tau}, \ \ 0 \le \tau \le 1, \\ &\text{s.t.} \quad \boldsymbol{X}(\boldsymbol{w}_+ - \boldsymbol{w}_-) = \boldsymbol{y} \quad \text{and} \quad \boldsymbol{w}_+, \boldsymbol{w}_- \ge \boldsymbol{0}\,. \end{aligned}$$

Imposing the constraints using a set of Lagrange multipliers $\boldsymbol{\nu}_+, \boldsymbol{\nu}_- \ge \boldsymbol{0}$ and $\lambda \in \mathbb{R}$, we have

$$\min_{\boldsymbol{w}} \sup_{\boldsymbol{\nu}_+, \boldsymbol{\nu}_- \ge \boldsymbol{0}, \lambda} \left\{ \|\boldsymbol{w}_+ - \boldsymbol{w}_-\|_{2-\tau}^{2-\tau} + \boldsymbol{\lambda}^\top \Big(\boldsymbol{X}(\boldsymbol{w}_+ - \boldsymbol{w}_-) - \boldsymbol{y} \Big) - \boldsymbol{w}_+^\top \boldsymbol{\nu}_+ - \boldsymbol{w}_-^\top \boldsymbol{\nu}_- \right\}.$$

The set of KKT conditions are

$$\begin{cases} \boldsymbol{w}_+, \boldsymbol{w}_- \ge \boldsymbol{0}\,, \\ \boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}\,, \\ +\operatorname{sign}(\boldsymbol{w}) \odot |\boldsymbol{w}|^{\odot(1-\tau)} - \boldsymbol{X}^\top \boldsymbol{\lambda} \succcurlyeq \boldsymbol{0}\,, \\ -\operatorname{sign}(\boldsymbol{w}) \odot |\boldsymbol{w}|^{\odot(1-\tau)} + \boldsymbol{X}^\top \boldsymbol{\lambda} \succcurlyeq \boldsymbol{0}\,, \\ \Big(\operatorname{sign}(\boldsymbol{w}) \odot |\boldsymbol{w}|^{\odot(1-\tau)} - \boldsymbol{X}^\top \boldsymbol{\lambda} \Big) \odot \boldsymbol{w}_+ = \boldsymbol{0}\,, \\ \Big(\operatorname{sign}(\boldsymbol{w}) \odot |\boldsymbol{w}|^{\odot(1-\tau)} - \boldsymbol{X}^\top \boldsymbol{\lambda} \Big) \odot \boldsymbol{w}_- = \boldsymbol{0}\,, \end{cases}$$

where $\boldsymbol{w} = \boldsymbol{w}_+ - \boldsymbol{w}_-$. The first condition is imposed by the form of the updates (3.28) and the second condition is satisfied by the assumption at $t \to \infty$.

Using $\boldsymbol{w}_0 = \alpha\mathbf{1}$ with $\alpha \to 0$, we have

$$\begin{aligned}
\boldsymbol{w}_+(t) &= \exp_\tau\Big(-\frac{1}{1-\tau} - \eta\int_0^t \boldsymbol{X}^\top\boldsymbol{\delta}(z)\,\mathrm{d}z\Big) \\
&= \Big[-(1-\tau)\,\eta\,\boldsymbol{X}\int_0^t \boldsymbol{\delta}(z)\Big]_+^{\odot\frac{1}{1-\tau}}, \\
\boldsymbol{w}_-(t) &= \exp_\tau\Big(-\frac{1}{1-\tau} + \eta\int_0^t \boldsymbol{X}^\top\boldsymbol{\delta}(z)\,\mathrm{d}z\Big) \\
&= \Big[+(1-\tau)\,\eta\,\boldsymbol{X}\int_0^t \boldsymbol{\delta}(z)\Big]_+^{\odot\frac{1}{1-\tau}}.
\end{aligned}$$

Setting $\boldsymbol{\lambda} = -(1-\tau)\,\eta\int_0^\infty \boldsymbol{\delta}(z)$ satisfies the remaining KKT conditions. □

Corollary 7. *Under the assumptions of Theorem 6, the reparameterized tempered EGU$^\pm$ updates* (3.30) *also recover the minimum* $\mathrm{L}_{2-\tau}$*-norm solution where* $\boldsymbol{w}(t) = q_\tau(\boldsymbol{u}_+(t)) - q_\tau(\boldsymbol{u}_-(t))$.

3.6.2 Partial Results on the Matrices

The linear regression problem can be generalized to the matrix case where the inputs $\boldsymbol{X}_n$ are symmetric matrices and the goal is to learn a symmetric weight matrix $\boldsymbol{W} \in \mathbb{S}^d$ which minimizes the squared error

$$\min_{\boldsymbol{W}\in\mathbb{S}^d} \tfrac{1}{2}\,\|\boldsymbol{X}(\boldsymbol{W}) - \boldsymbol{y}\|^2,$$

in which $\boldsymbol{X}(\boldsymbol{W})_n \doteq \operatorname{tr}\big(\boldsymbol{X}_n\boldsymbol{W}\big)$.

The generalization of Theorem 4 to symmetric matrices $\boldsymbol{W} \in \mathbb{S}^d$ requires a machinery to handle the matrix-matrix derivatives for the Jacobians. Instead, we first consider a natural extension of the reparameterized tempered EGU$^\pm$ to the

symmetric PSD matrices: given $\boldsymbol{U}_+, \boldsymbol{U}_- \in \mathbb{R}^{d\times d}$, we define

$$\begin{aligned} \boldsymbol{W}_+ &= \Big(\frac{1}{2-\tau}\Big)^{-\frac{1}{1-\tau}} \big(\boldsymbol{U}_+\boldsymbol{U}_+^\top\big)^{\frac{1}{2-\tau}}, \\ \boldsymbol{W}_- &= \Big(\frac{1}{2-\tau}\Big)^{-\frac{1}{1-\tau}} \big(\boldsymbol{U}_-\boldsymbol{U}_-^\top\big)^{\frac{1}{2-\tau}}, \end{aligned} \tag{3.32}$$

and set $\boldsymbol{W} = \boldsymbol{W}_+ - \boldsymbol{W}_-$. The solution for the linear regression problem using the reparameterization (3.32) can be obtained by solving

$$\begin{aligned} \dot{\boldsymbol{U}}_+ &= -\eta\, \boldsymbol{X}^*(\boldsymbol{\delta})\, (\boldsymbol{U}_+\boldsymbol{U}_+)^{-\frac{1-\tau}{2-\tau}}\, \boldsymbol{U}_+, \\ \dot{\boldsymbol{W}_+^{2-\tau}} &= \dot{\boldsymbol{U}}_+\boldsymbol{U}_+^\top + \boldsymbol{U}_+\dot{\boldsymbol{U}}_+^\top \\ &= -\eta \left(\boldsymbol{X}^*(\boldsymbol{\delta})\boldsymbol{W}_+ + \boldsymbol{W}_+\, \boldsymbol{X}^*(\boldsymbol{\delta})\right), \end{aligned} \tag{3.33}$$

where $\boldsymbol{\delta}(t) = \boldsymbol{X}(\boldsymbol{W}(t)) - \boldsymbol{y}$ and $\boldsymbol{X}^*(\boldsymbol{v}) = \sum_n \boldsymbol{X}_n v_n$. Also, the constant factors are absorbed into the learning rate. A similar set of equations hold for $\boldsymbol{W}_-$ with a sign flip. Solving for $\boldsymbol{W}_+$ and $\boldsymbol{W}_-$ yields

$$\begin{aligned} \boldsymbol{W}_+(t) &= \Big[\boldsymbol{W}_0^{2-\tau} - \eta \int_s^t \boldsymbol{S}\big(\boldsymbol{\delta}(z), \boldsymbol{W}_+(z)\big)\, \mathrm{d}z\Big]_+^{\frac{1}{2-\tau}}, \\ \boldsymbol{W}_-(t) &= \Big[\boldsymbol{W}_0^{2-\tau} + \eta \int_s^t \boldsymbol{S}\big(\boldsymbol{\delta}(z), \boldsymbol{W}_-(z)\big)\, \mathrm{d}z\Big]_+^{\frac{1}{2-\tau}}. \end{aligned} \tag{3.34}$$

where $\boldsymbol{S}(\boldsymbol{\delta}, \boldsymbol{W}_\pm) = \boldsymbol{X}^*(\boldsymbol{\delta})\boldsymbol{W}_\pm + \boldsymbol{W}_\pm\boldsymbol{X}^*(\boldsymbol{\delta})$. Update (3.34) is a generalization of the update given in [31], which is obtained by setting $\tau = 1$. We generalize the results of [31] as follows: In the special case where the inputs $\boldsymbol{X}_n$ are symmetric and commutative, i.e., $\boldsymbol{X}_n\boldsymbol{X}_{n'} = \boldsymbol{X}_{n'}\boldsymbol{X}_n$ for all n, n', the following result holds.

Proposition 6. *Given that the inputs $\boldsymbol{X}_n$ to the matrix linear regression problem are symmetric and commutative, if $\boldsymbol{X}(\boldsymbol{W}(\infty)) = \boldsymbol{y}$ holds, the reparameterized update* (3.34) *converges to the minimum* $\mathrm{L}_{2-\tau}$ *matrix norm solution when* $\boldsymbol{W}_0 = \lim_{\alpha\to 0_+} \alpha\boldsymbol{I}$.

Proof. Note that the optimization problem corresponds to

$$\min_{\boldsymbol{W}_+, \boldsymbol{W}_-} \|\boldsymbol{W}_+ - \boldsymbol{W}_-\|_{2-\tau}^{2-\tau}, \quad \text{s.t.} \quad \boldsymbol{X}(\boldsymbol{W}_+ - \boldsymbol{W}_-) = \boldsymbol{y}\,.$$

To prove the proposition, it suffices to show that the solution satisfies the following KKT conditions:

$$\begin{cases} \boldsymbol{W}_+, \boldsymbol{W}_- \succcurlyeq 0 \\ \boldsymbol{X}(\boldsymbol{W}) = \boldsymbol{y} \\ \boldsymbol{X}^*(\boldsymbol{\nu}) \preccurlyeq \mathbf{sign}\Big(\boldsymbol{W}\Big)\,|\boldsymbol{W}|^{1-\tau} \\ \mathbf{sign}\Big(\boldsymbol{W}\Big)\,|\boldsymbol{W}|^{1-\tau} \preccurlyeq \boldsymbol{X}^*(\boldsymbol{\nu}) \\ \Big(\mathbf{sign}\Big(\boldsymbol{W}\Big)\,|\boldsymbol{W}|^{1-\tau} - \boldsymbol{X}^*(\boldsymbol{\nu})\Big)\,\boldsymbol{W}_+ = \mathbf{0} \\ \Big(\mathbf{sign}\Big(\boldsymbol{W}\Big)\,|\boldsymbol{W}|^{1-\tau} - \boldsymbol{X}^*(\boldsymbol{\nu})\Big)\,\boldsymbol{W}_- = \mathbf{0} \end{cases}$$

for some $\boldsymbol{\nu} \in \mathbb{R}^n$. Also, $\boldsymbol{W} = \boldsymbol{W}_+ - \boldsymbol{W}-$ and $\mathbf{sign}(\boldsymbol{A}) = \mathbf{sign}(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top) = \boldsymbol{V}\mathbf{sign}(\boldsymbol{D})\boldsymbol{V}^\top$ is the sign function applied to the eigenvalues. The first condition is satisfied by the form of the updates. The second condition follows from the assumption. To show the remaining conditions, first note that the symmetric and commutative inputs $\boldsymbol{X}_n$ are diagonalizable, thus the eigenvectors of $\boldsymbol{X}^*(\boldsymbol{v})$ are fixed and the eigenvalues can be written as $\lambda_i(\boldsymbol{v})\,,$ for $i \in [d]$.

In the limit $\boldsymbol{W}_0 = \lim_{\alpha \to 0_+} \alpha \boldsymbol{I}$, the reparameterization updates (3.34) can be written as

$$\begin{aligned} \boldsymbol{W}_+(t) &= \Big[-\eta \int_0^t \boldsymbol{S}\Big(\boldsymbol{\delta}(z), \boldsymbol{W}_+(z)\Big)\,\mathrm{d}z \Big]_+^{\frac{1}{2-\tau}}, \\ \boldsymbol{W}_-(t) &= \Big[+\eta \int_0^t \boldsymbol{S}\Big(\boldsymbol{\delta}(z), \boldsymbol{W}_-(z)\Big)\,\mathrm{d}z \Big]_+^{\frac{1}{2-\tau}}, \end{aligned}$$

where $[\boldsymbol{A}]_+ = [\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top]_+ = \boldsymbol{V}[\boldsymbol{D}]_+\boldsymbol{V}^\top$ is applied to the eigenvalues. Since both $\boldsymbol{W}_+(t)$ and $\boldsymbol{W}_-(t)$ share the same eigenvectors as $\boldsymbol{X}^*(\boldsymbol{\nu})$ for all $\boldsymbol{\nu}$, it suffices to show that the eigenvalues satisfy the complementary slackness and dual feasibility KKT conditions. Let $\omega_{+,i}(t)$ and $\lambda_i(t)$ denote the i-th eigenvector of $\boldsymbol{W}_+(t)$ and $\boldsymbol{X}^*(\boldsymbol{u}(t))$, respectively. For fixed eigenvectors, Equation (3.33) imposes the following differential equation on the eigenvalues

$$\frac{\partial}{\partial t}\,\omega_{+,i}^{2-\tau} = 2-\tau\,\omega_{+,i}^{1-\tau}\,\dot{\omega}_{+,i} = -\eta\left(\lambda_i\,\omega_{+,i} + \omega_{+,i}\,\lambda_i\right) = -2\,\eta\,\omega_{+,i}\,\lambda_i\,.$$

A similar ODE holds for $\omega_{-,i}(t)$ with a sign flip. Imposing the constraint $\omega_{+,i} \geq 0$ with the boundary condition $\omega_{+,i}(s) = 0$ yields

$$\begin{aligned}\omega_{+,i}(t) &= \Big[-\frac{2\,\eta}{2-\tau}\int_0^t \lambda_i(z)\,\mathrm{d}z\Big]_+^{\frac{1}{1-\tau}}\,,\\ \omega_{-,i}(t) &= \Big[+\frac{2\,\eta}{2-\tau}\int_0^t \lambda_i(z)\,\mathrm{d}z\Big]_+^{\frac{1}{1-\tau}}\,.\end{aligned}$$

Thus, the updates can be written

$$\begin{aligned}\boldsymbol{W}_+(t)^{1-\tau} &= \Big[-c\int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\,\mathrm{d}z\Big]_+\,,\\ \boldsymbol{W}_-(t)^{1-\tau} &= \Big[+c\int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\,\mathrm{d}z\Big]_+\,,\end{aligned} \tag{3.35}$$

for some constant $c > 0$. This proves the existence of $\boldsymbol{\nu}$ so that the KKT conditions are satisfied. The proof for the case $\tau = 1$ follows similarly to [31] using a limit argument. □

Next, we provide an alternative dynamic based on generalizing the tempered Bregman divergence (3.25) to the positive semi-definite (PSD) matrices. Let us first introduce the matrix $\mathbf{log}_\tau$ operator as follows:

$$\mathbf{log}_\tau \boldsymbol{A} = \boldsymbol{V}\mathbf{log}_\tau(\boldsymbol{D})\boldsymbol{V}^\top\,,$$

where $\mathbf{log}_\tau(\boldsymbol{D})$ is the diagonal matrix for which the $\log_\tau$ function is applied to the diagonal elements. The matrix tempered Bregman divergence can be defined as

$$D_{F_\tau}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \operatorname{tr}\left(\widetilde{\boldsymbol{W}}\,\mathbf{log}_\tau\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\,\mathbf{log}_\tau\boldsymbol{W} - \frac{1}{2-\tau}\left(\widetilde{\boldsymbol{W}}^{2-\tau} - \boldsymbol{W}^{2-\tau}\right)\right). \quad (3.36)$$

The continuous-time learning problem for a general symmetric matrix $\boldsymbol{W}(t)$ can again be achieved via introducing a pair of PSD matrices $\boldsymbol{W}_+(t), \boldsymbol{W}_-(t) \succcurlyeq \mathbf{0}$ such that $\boldsymbol{W}_+(0) = \boldsymbol{W}_-(0) = \boldsymbol{W}_0$ and $\boldsymbol{W}(t) = \boldsymbol{W}_+(t) - \boldsymbol{W}_-(t)$. Following a similar derivation as in the vector case (3.26) and imposing the positive semi-definiteness constraints in the form of Lagrange multipliers $\boldsymbol{\lambda}_+, \boldsymbol{\lambda}_- \geq \mathbf{0}$ applied to the eigenvalues of $\boldsymbol{W}_+, \boldsymbol{W}_-$, we have

$$\begin{aligned}\dot{\mathbf{log}}_\tau\boldsymbol{W}_+ &= -\eta\nabla_{\boldsymbol{W}}L(\boldsymbol{W}) + \boldsymbol{V}_+\operatorname{diag}(\boldsymbol{\lambda}_+)\boldsymbol{V}_+^\top,\\ \dot{\mathbf{log}}_\tau\boldsymbol{W}_- &= +\eta\nabla_{\boldsymbol{W}}L(\boldsymbol{W}) + \boldsymbol{V}_-\operatorname{diag}(\boldsymbol{\lambda}_-)\boldsymbol{V}_-^\top,\end{aligned} \quad (3.37)$$

where $\boldsymbol{V}_\pm$ denotes the matrix of eigenvectors of $\boldsymbol{W}_\pm$. Integrating the r.h.s. yields the following update equations

$$\begin{aligned}\boldsymbol{W}_+(t) &= \mathbf{exp}_\tau\Big(\mathbf{log}_\tau\boldsymbol{W}_0 - \eta\int_0^t\nabla_{\boldsymbol{W}}L(\boldsymbol{W}(z))\,\mathrm{d}z\Big),\\ \boldsymbol{W}_-(t) &= \mathbf{exp}_\tau\Big(\mathbf{log}_\tau\boldsymbol{W}_0 + \eta\int_0^t\nabla_{\boldsymbol{W}}L(\boldsymbol{W}(z))\,\mathrm{d}z\Big),\end{aligned} \quad (3.38)$$

where matrix $\mathbf{exp}_\tau$ is defined similarly in terms of the eigen-decomposition of the input.

However, the dynamic induced by the tempered Bregman divergence in (3.37) is different than the one obtained from the reparameterization (3.32). As an

example, consider the matrix EGU$^{\pm}$ case for $\tau = 1$. The GD update on $\boldsymbol{U}_+$ yields

$$\begin{aligned}
\dot{\boldsymbol{U}}_+ &= -\nabla_{\boldsymbol{W}} L(\boldsymbol{W})\, \boldsymbol{U}_+ \,, \\
\dot{\boldsymbol{W}}_+ &= \dot{\boldsymbol{U}}_+ \boldsymbol{U}_+^\top + \boldsymbol{U}_+ \dot{\boldsymbol{U}}_+^\top \\
&= -\Big(\boldsymbol{W}_+ \, \nabla_{\boldsymbol{W}} L(\boldsymbol{W}) + \nabla_{\boldsymbol{W}} L(\boldsymbol{W}) \boldsymbol{W}_+ \Big) \,,
\end{aligned}$$

which is different than the dynamic given in (3.37) for $\tau = 1$. (The dynamic for $\boldsymbol{W}_-$ follows similarly with a sign flip.) Nevertheless, we show that for the underdetermined linear regression problem with commutative inputs, the tempered Bregman updates (3.38) yield the same minimum-norm solution in the limit when the initial solution goes to zero.

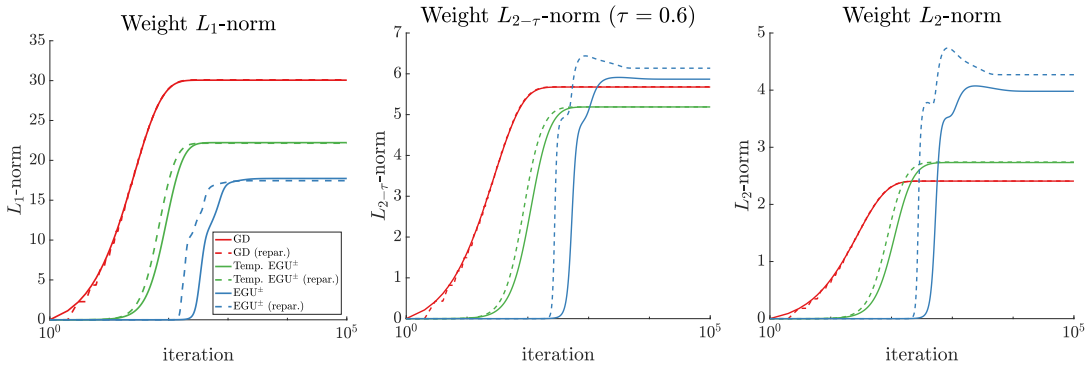
Substituting for the gradients in (3.38) yields

$$\begin{aligned}
\boldsymbol{W}_+(t) &= \mathbf{exp}_\tau \Big(\mathbf{log}_\tau \boldsymbol{W}_0 - \eta \int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\, \mathrm{d}z \Big), \\
\boldsymbol{W}_-(t) &= \mathbf{exp}_\tau \Big(\mathbf{log}_\tau \boldsymbol{W}_0 + \eta \int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\, \mathrm{d}z \Big),
\end{aligned} \tag{3.39}$$

where $\boldsymbol{\delta}(t) = \boldsymbol{X}(\boldsymbol{W}(t)) - \boldsymbol{y}$ and $\boldsymbol{X}^*(\boldsymbol{v}) = \sum_n \boldsymbol{X}_n v_n$.

Proposition 7. *Given that the inputs $\boldsymbol{X}_n$ to the matrix linear regression problem are symmetric and commutative, if $\boldsymbol{X}(\boldsymbol{W}(\infty)) = \boldsymbol{y}$ holds, the tempered matrix EGU$^{\pm}$ update* (3.39) *converges to the minimum $\mathrm{L}_{2-\tau}$ matrix norm solution when $\boldsymbol{W}_0 = \lim_{\alpha \to 0_+} \alpha \boldsymbol{I}$.*

Proof. The first KKT condition is satisfied by the form of the updates. The second condition follows from the assumption. To show the remaining conditions, note that the matrix tempered Bregman updates (3.39) in the limit $\boldsymbol{W}_0 = \lim_{\alpha \to 0_+} \alpha \boldsymbol{I}$



(a) (b) (c)

Figure 3.3: Underdetermined linear regression: (a)-(c) norms of the solutions obtained using GD, EGU$^\pm$, and tempered EGU$^\pm$ ($\tau = 0.6$) along with their reparameterized forms on an underdetermined linear regression problem.

can be written as

$$\begin{aligned} \boldsymbol{W}_+(t) &= \Big[-(1-\tau)\,\eta \int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\,\mathrm{d}z \Big]_+^{\frac{1}{1-\tau}}, \\ \boldsymbol{W}_-(t) &= \Big[+(1-\tau)\,\eta \int_0^t \boldsymbol{X}^*(\boldsymbol{\delta}(z))\,\mathrm{d}z \Big]_+^{\frac{1}{1-\tau}}. \end{aligned} \tag{3.40}$$

Thus, $\boldsymbol{W}_+(t), \boldsymbol{W}_-(t)$ share the same eigenvectors as $\boldsymbol{X}^*$. Setting $\boldsymbol{\nu} = -(1-\tau)\,\eta \int_0^\infty \boldsymbol{\delta}(z)\,\mathrm{d}z$ satisfies the remaining conditions. The case $\tau = 1$ also follows by a limit argument, similar to the one given in [31]. □

Note that the analysis for the tempered updates holds in the continuous domain. Moreover, the analysis is based on the assumption that the initialization tends to zero. In practice, however, the learning proceeds in steps and the initial solution cannot be made arbitrary small. Therefore, such assumptions may seem impractical. Nevertheless, we show experimentally that, despite the approximations, the discrete updates behave very closely to their continuous-time counterparts.

3.7 Experiments

In this section, we first consider the GD, EGU$^{\pm}$, and tempered EGU$^{\pm}$ updates and their reparameterizations for an underdetermined linear regression problem. Next, we provide preliminary results on reparameterizing deep neural networks.

3.7.1 Minimum-norm Solutions for Linear Regression

We apply the GD, EGU$^{\pm}$, and the tempered EGU$^{\pm}$ ($\tau = 0.6$) updates on a toy underdetermined linear regression problem. For the experiment, we consider $N = 100$ samples of $d = 200$ dimensional zero-mean unit-variance Gaussian inputs and a linear target obtained with a random Gaussian weight. We apply the updates as well as their reparameterized forms using the weights $\boldsymbol{u}_+, \boldsymbol{u}_- \in \mathbb{R}^d$. We consider a small learning rate of $\eta = 0.05$ and set the initial solutions $\boldsymbol{w}_+(0) = \boldsymbol{w}_-(0) = 1\text{e}{-5} \times \mathbf{1}$. This corresponds to a zero initial solution for the actual weights $\boldsymbol{w}(0) = \boldsymbol{w}_+(0) - \boldsymbol{w}_-(0) = \mathbf{0}$. We apply the updates on the full batch for $1\text{e}+5$ iterations. The results are shown in Figure 3.3(a)-(c). As expected, EGU$^{\pm}$ and GD converge to the minimum L_1 and L_2-norm solutions, respectively. This is also verified by solving for the minimum L_1 and L_2-norm solutions numerically. Additionally, tempered EGU$^{\pm}$ achieves the minimum $\text{L}_{2-\tau}$-norm. Finally, note that the reparameterized versions of the algorithms track the original algorithms very closely.

3.7.2 Reparameterizing Weights of Neural Networks

We consider a convolutional neural network for classifying the MNIST handwritten digits dataset [44]. The network consists of two convolutional layers of size 32 and 64, followed by two fully connected layers of size 1024 and 10. We

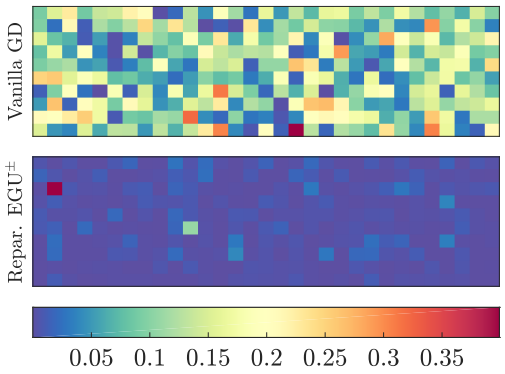


Figure 3.4: Absolute values of (a slice of) the weights of the last layer for the vanilla GD and the reparameterized EGU$^{\pm}$ networks. The L_1-norm of the GD weights is 571.1 while for reparameterized EGU$^{\pm}$, the L_1-norm is 176.7.

train the first network normally for 100 epochs using SGD. The top-1 accuracy obtained on the test set is 99.13%. Next, we train a second network similarly, except we apply the EGU$^{\pm}$ reparameterization to the weights of the final layer (size 1024×10) where we expect a high amount of redundancy due to the large size of the network. As a result, the reparameterized weights naturally exhibit a high amount of sparsity compared to the weights of the vanilla network, as shown in Figure 3.4. As can be seen, most components of the reparameterized weights are concentrated around zero and only a small fraction of the components have large values. In fact, the L_1-norm of the GD weights is 571.1 while for reparameterized EGU$^{\pm}$, the L_1-norm is 176.7. To test the significance of the small weights, we clamp the weights of the final layer for which the absolute values are below a certain threshold, to zero during the inference. The top-1 test accuracy results are shown in Table 3.2 for different values of the threshold and the levels of sparsity achieved. Note that even with 98.90% sparsity, the reparameterized EGU$^{\pm}$ network achieves 97.48% test accuracy.

Threshold	0	1e−3	5e−3	1e−2	5e−2	1e−1
Sparsity (%)	00.00	25.58	62.02	79.08	97.55	98.90
Accuracy (%)	99.02	99.02	99.00	98.96	98.20	97.48

Table 3.2: Different levels of sparsity achieved by thresholding the weights of the reparameterized last layer and the corresponding top-1 test set accuracy. Even with 98.90% sparsity the network achieves 97.48% test accuracy.

3.8 Discussion

In this chapter, we discussed the continuous-time mirror descent updates and provided a general framework for reparameterizing these updates. Additionally, we introduced the tempered EGU$^{\pm}$ updates and their reparameterized forms. The tempered EGU$^{\pm}$ updates include the two commonly used updates, namely, gradient descent and exponentiated gradient updates, and interpolate between the two updates. We showed that under certain conditions for the underdetermined linear regression problem, the tempered EGU$^{\pm}$ updates converge to the minimum $L_{2-\tau}$-norm solution. Finally, we expanded the reparameterized updates to the matrix case, generalizing the results of [31].

The current work leads to many interesting future directions:

- The reparameterization equivalence theorem holds only in the continuous-time. Clearly, the equivalence relation breaks down after discretization, however, the discretized reparameterized updates seem to track the original updates in many importnat cases (as we show for one of the cases in the next chapter). A key research direction is to find general conditions under which the discretized updates closely track the original continuous-time counterparts.

- A more general treatment of the underdetermined linear regression requires analyzing the results for arbitrary start vectors. Also, extending the matrix

results to non-commutative matrices (as discussed in [31]) is still an open problem. Furthermore, developing a matrix form of the reparameterization theorem is left for future work.

- Perhaps the most important application of the current work is reparameterizing the weights of deep neural networks for achieving sparse solutions or obtaining an implicit form of regularization that mimics a trade-off between the ridge and lasso methods (e.g. elastic net regularization [76]).

Chapter 4

Winnowing with Gradient Descent

4.1 Introduction

Multiplicative updates started with the Winnow algorithm for learning disjunctions and linear threshold functions [47, 36]. These algorithms updates their weights by multiplicative factors and often learn sparse targets with a logarithmic dependence on the number of features. For example, Winnow makes at most $k \log n$ mistakes on the sequence of n-dimensional boolean vectors labeled consistently with a k-literal disjunction.

Another paradigmatic case is the so called "expert setting" where the learner is to perform as well as a single feature/expert. For sequences of n-dimensional feature vectors with one consistent feature, the total loss of the update is typically $\mathcal{O}(\log n)$. The situation is repeated for linear regression. When the weight vector realizing the labels is sparse, then the normalized and unnormalized EG algorithms incur loss at most $\mathcal{O}(\log n)$. So far we focused on the noise-free case in our

discussion. In the noisy case (when the solution has non-zero loss), then additional square root terms appear in the regret bounds of these algorithms.

We first observe that in continuous-time, multiplicative updates can be exactly reparameterized as continuous gradient descent. This is done by replacing the linear weights w_i by u_i^2 and applying continuous-time GD w.r.t. the new parameter u_i (See Figure 4.1). We show how the discretization of the continuous multiplicative updates results in the usual Winnow and EG updates. More importantly, the discretizations of the reparameterized continuous multiplicative updates results in discrete time GD variants of these updates. However, the equivalence between the multiplicative updates and the gradient descent reparameterizations does not hold after discretization. Nevertheless, we show that the GD reparameterizations closely track the multiplicative originals by proving the same regret bounds for them as were known for the originals. In each case, the constants we obtain for the second order terms are the same or slightly different in either direction.

For the discrete GD updates, lower-bounds that grow linearly in the number of features n are known for all three settings [39, 67]. In these lower-bounds, the instances are essentially the rows of an $n \times n$ Hadamard matrix and the target is in the simplest case a single column of this matrix. When averaged over targets, then for linear regression the lower-bound holds even if the input vectors $\boldsymbol{x}$ are replaced by a feature map $\phi(\boldsymbol{x})$ of any kernel [67]. It was conjectured in [25] that the linear lower bound holds for every neural network, as long as it is trained with gradient descent.

For a long time, the Hadamard problem remained a case which highlighted the fundamental difference between the GD and EGU updates. Specifically, a single linear neuron (with any kernelization) fails to learn the Hadamard problem via backpropagation. Although EGU can solve this problem efficiently, it was

conjectured that there would be no neuron that could simulate the multiplicative updates of EGU via backpropagation. This conjecture is now contradicted by the linear network in Figure 4.1. If this network is trained with GD, then it experimentally solves the Hadamard problem as efficiently as EG and EGU. Thus for the Hadamard problem, any kernel method satisfies the linear lower-bound while there is a simple two-layer linear network that, when trained with GD, can realize the $\log n$ dependence on the number of examples. Also experimentally, when all the missing $n^2 - n$ connections are added at the bottom layer and initialized to zero, then the now fully-connected two-layer linear network again incurs the linear lower-bound when trained with GD.

Previous work. Regret bounds for multiplicative updates go back to the Winnow [47], Weighted Majority algorithm [40], and Aggregating [68] algorithms. A key insight made by [31] (followed by [73] and [64]) showed that when the weights w_i of a linear predictor are replaced by u_i^2, then the continuous-time gradient descent update on u_i's imposes an implicit regularization on w_i's that makes the reparameterized weight vector $\boldsymbol{w}$ converge to the minimum L_1-norm solution[1]. The EG updates are known to connect with the L_1-norm regularization. Indeed, the relative entropy, which is the divergence motivating the EG updates, is strongly convex w.r.t. the L_1-norm [59]. We show that the continuous-time unnormalized exponentiated gradient algorithm is in fact equivalent to the continuous GD on u_i's. Curiously enough, this reparameterization method has been used by game theorists to convert problems defined on the simplex to the unit sphere [2]. In particular, the replicator dynamics of evolutionary game theory corresponds to EG and gradient descent equivalents have been investigated for this dynamic. The equivalence relation between the EG updates and the cor-

[1]Note that [31] did the reparameterization already in the much richer matrix context. For the sake of simplicity we focus in this chapter on the diagonal case

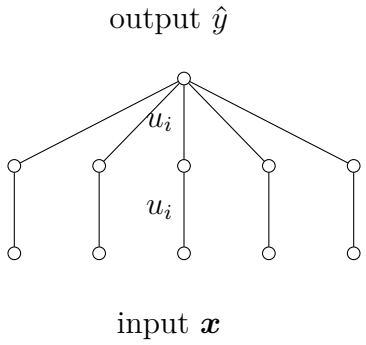


Figure 4.1: Reparameterizing theweights w_i of a linear neuron by u_i^2.

responding reparameterized updates holds trivially in the continuous case. The main result of this chapter is showing that essentially the same regret bounds hold for the discretized GD reparameterizations of the multiplicative updates.

Note that obtaining an online regret bound is one of the most stringent learning criterion since it must hold for worst-case sequences of examples. Standard conversions exist to randomized settings (See e.g. [40]). Also, EGU is a special case of mirror descent where the link function is the component-wise logarithm. A more general framework for reparameterizing mirror descent was developed in Chapter 3.

Outline. In the next section, we show the equivalence between the continuous-time exponentiated gradient updates and their reparameterized gradient descent versions. We also motivate the discretizations of the continuous updates. In the following three chapters, we reprove the regret bounds for the discrete gradient descent variants of Winnow, the Hedge algorithm, and the EG algorithms for linear regression. Lower-bounds and some implications of neural network training are briefly discussed in Section 4.6. We conclude with a number of open problems. The material in chapter appeared in COLT 2020 [9].

4.2 Reparameterizing the Continuous-time Exponentiated Gradient Algorithms

In the following, we discuss the continuous-time EGU and EG updates and derive their reparameterized forms. Although reparamterization method presented here is more versatile and applies to a wider class of continuous mirror descent updates (as shown in Chapter 3), we only focus on these two known cases in this chapter [2, 57].

The continuous-time EGU update can be seen as the solution of the following ordinary differential equation (ODE) defined on $\mathbb{R}^n_{\geq 0}$ with the boundary condition $\boldsymbol{w}(t=0) = \boldsymbol{w}^0 \in \mathbb{R}^n_{\geq 0}$:

$$\dot{\log}\, \boldsymbol{w}(t) = -\eta\, \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}(t))\,, \tag{4.1}$$

in which the $\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}(t))$ denotes the gradient of the loss wrt $\boldsymbol{w}$ evaluated at $\boldsymbol{w}(t)$. Typically, the loss $\ell(\,\cdot\,)$ also depends on a given example $\boldsymbol{x}(t)$ and possibly a label $y(t)$. Although the dynamic of the continuous-time EGU (4.1) is fundamentally different than a dynamic based on GD, we apply the results of Chapter 3 to cast this update as GD. Namely, we substitute $\boldsymbol{w}(t) = q(\boldsymbol{u}(t)) \doteq \boldsymbol{u}(t) \odot \boldsymbol{u}(t)$ where the new parameter $\boldsymbol{u}(t) \in \mathbb{R}^n$ is updated via GD on the composite loss $\ell \circ q\,(\cdot)$.

Remark 1 (Reparameterized continuous-time EGU)**.** *The solution of the ODE* (4.1) *is equal to the* $\boldsymbol{w}(t) = q(\boldsymbol{u}(t))$ *for all* $t \geq 0$*, where* $q(\boldsymbol{u}) := \boldsymbol{u} \odot \boldsymbol{u}$ *and the new parameter* $\boldsymbol{u}(t)$ *is the solution of the following ODE defined on* $\mathbb{R}^n$*:*

$$\dot{\boldsymbol{u}}(t) = -\eta/4\, \nabla_{\boldsymbol{u}}\, \ell \circ q\, \Big(\boldsymbol{u}(t)\Big), \text{ with initial condition } \boldsymbol{u}(t=0) = \boldsymbol{u}^0 \in \mathbb{R}^n. \tag{4.2}$$

See Example 3 for a proof.

The discrete-time EGU can be derived as the finite difference approximation

of (4.1) with a step-size of one, that is,

$$\log \boldsymbol{w}^{t+1} - \log \boldsymbol{w}^t = -\eta \, \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}^t) \,, \tag{4.3}$$

which corresponds to the solution to the following objective function:

$$\boldsymbol{w}^{t+1} = \underset{\tilde{\boldsymbol{w}} \in \mathbb{R}^n_{\geq 0}}{\arg\min} \left\{ 1/\eta \, D_{\mathrm{RE}}(\tilde{\boldsymbol{w}}, \boldsymbol{w}^t) + \hat{\ell}(\tilde{\boldsymbol{w}} | \, \boldsymbol{w}^t) \right\}. \tag{4.4}$$

Here, $\hat{\ell}(\tilde{\boldsymbol{w}} | \, \boldsymbol{w}^t)$ denotes the first-order Taylor series approximation[2] of the loss $\ell(\tilde{\boldsymbol{w}})$ at $\boldsymbol{w}^t$

$$\hat{\ell}(\tilde{\boldsymbol{w}} | \, \boldsymbol{w}^t) = \ell(\tilde{\boldsymbol{w}}) + (\tilde{\boldsymbol{w}} - \boldsymbol{w}^t) \cdot \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}^t) \,.$$

Similarly, discretizing (4.2) yields the discrete-time reparameterized EGU update

$$\boldsymbol{u}^{t+1} - \boldsymbol{u}^t = -\eta/4 \, \nabla_{\boldsymbol{u}} \, \ell \circ q \left(\boldsymbol{u}^t \right) = -\eta/2 \; \boldsymbol{u}^t \odot \nabla_{\boldsymbol{w}} \, \ell \Big(\boldsymbol{w}^t \Big) \,, \tag{4.5}$$

with $\boldsymbol{w}^t = \boldsymbol{u}^t \odot \boldsymbol{u}^t$. This can be seen as the solution to the objective

$$\boldsymbol{u}^{t+1} = \underset{\boldsymbol{u}^{t+1} \in \mathbb{R}^n}{\arg\min} \left\{ 1/\eta \, \|\tilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + 1/2 \; \widehat{\ell \circ q} \left(\tilde{\boldsymbol{u}} \, | \, \boldsymbol{u}^t \right) \right\}, \tag{4.6}$$

where $\widehat{\ell \circ q} \left(\tilde{\boldsymbol{u}} \, | \, \boldsymbol{u}^t \right)$ is the first-order Taylor series approximation of $\ell \circ q \, (\tilde{\boldsymbol{u}})$ at $\boldsymbol{u}^t$:

$$\widehat{\ell \circ q} \left(\tilde{\boldsymbol{u}} \, | \, \boldsymbol{u}^t \right) = \ell \circ q \left(\boldsymbol{u}^t \right) + (\tilde{\boldsymbol{u}} - \boldsymbol{u}^t) \cdot \nabla_{\boldsymbol{u}} \, \ell \circ q \left(\boldsymbol{u}^t \right). \tag{4.7}$$

A similar continuous-time dynamic can be constructed for the normalized EG update and its equivalent reparameterized form. This involves applying projected gradient updates to maintain the constraint $\|\boldsymbol{w}(t)\|_1 = 1$ (respectively, $\|\boldsymbol{u}(t)\|_2^2 =$

[2]Note that a backward Euler approximation of (4.1) yields the *implicit form* of the update [41], which corresponds to using $\ell(\tilde{\boldsymbol{w}})$ instead of $\hat{\ell}(\tilde{\boldsymbol{w}} | \, \boldsymbol{w}^t)$ in (4.4).

1). Here, we state the result for the discretized reparameterized form and refer the reader to Chapter 3 for further details. Our construction here is based on directly discretizing the functional form, which follows the derivation of the discretized updates in Section 3.3. That is, adding the Lagrange multiplier λ to (4.6) to enforce the constraint $\left(\boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}\right) \cdot \mathbf{1} = 1$, we have

$$\boldsymbol{u}^{t+1} = \operatorname*{arg\,min}_{\boldsymbol{u}^{t+1} \in \mathbb{R}^n} \left\{ 1/\eta \, \|\tilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + 1/2 \, \widehat{\ell \circ q}\left(\tilde{\boldsymbol{u}} \,|\, \boldsymbol{u}^t\right) + \lambda \left(\left(\tilde{\boldsymbol{u}} \odot \tilde{\boldsymbol{u}}\right) \cdot \mathbf{1} - 1 \right) \right\} \quad (4.8)$$

which results in the reparameterized EG update

$$\boldsymbol{u}^{t+1} = \frac{\boldsymbol{u}^t - \eta/2 \, \boldsymbol{u}^t \odot \nabla_{\boldsymbol{w}} \, \ell(\boldsymbol{w}^t)}{\left\| \boldsymbol{u}^t - \eta/2 \, \boldsymbol{u}^t \odot \nabla_{\boldsymbol{w}} \, \ell(\boldsymbol{w}^t) \right\|_2}, \quad \text{where} \quad \boldsymbol{w}^t = \boldsymbol{u}^t \odot \boldsymbol{u}^t \,. \quad (4.9)$$

Clearly, after the discretization, the EGU and EG updates and their reparameterized forms are no longer equivalent. The key question that naturally arises is the following: *How well does the discretized reparameterized EGU update (and its normalized form) approximate the original EGU (respectively, the original EG) update?* In the following, we address this question by comparing the worst-case regrets of the original (un)normalized EG and its reparameterizations on three problems, namely, the Winnow algorithm for binary classification, the Hedge algorithm for the expert setting, and EGU and EG for linear regression. The following regret bounds for the reparameterizations are proven by bounding the progress towards a comparator. Curiously enough, the progress is measured i.t.o. the relative entropy divergence even though the reparameterized updates are motivated by regularizing with the squared Euclidean distance.

4.3 Reparameterization of the Winnow

The Winnow algorithm learns a linear threshold function for the task of binary classification. It is a special case of EGU when the loss is the hinge loss. The hinge loss on example $(\boldsymbol{x}^t, y^t)$ where $y^t \in \{\pm 1\}$ is defined as $\ell_H(\boldsymbol{w}|\, \boldsymbol{x}^t, y^t) = \Big[- y^t \left(\boldsymbol{w} \cdot \boldsymbol{x}^t - \theta\right)\Big]_+$. GD w.r.t. same loss results in the Perceptron update. The Winnow algorithm and its reparameterized form are given in the following. Both algorithms update their weights only when a mistake occurs because when the prediction is correct, then the gradient of the hinge loss is zero. See e.g. [39] for a comparative study.

Algorithm 1 Winnow Algorithm

Parameters initial weight $w^1 > 0$, learning rate η, threshold $\theta > 0$
Initialize $\boldsymbol{w}^1 = w^1\, \mathbf{1}_n$
for $t = 1$ to T **do**
  Receive instance $\boldsymbol{x}^t \in [0,1]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} +1 & \text{if } \boldsymbol{w}^t \cdot \boldsymbol{x}^t \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

  Receive label y^t and update

$$\boldsymbol{w}^{t+1} = \begin{cases} \boldsymbol{w}^t & \text{if } \hat{y}^t = y^t \\ \boldsymbol{w}^t \odot \exp(\eta\, y^t\, \boldsymbol{x}^t) & \text{otherwise} \end{cases}$$

Algorithm 2 Reparameterized Winnow

Parameters initial weight $u^1 \in \mathbb{R}$, learning rate η, threshold $\theta > 0$
Initialize $\boldsymbol{u}^1 = u^1\, \mathbf{1}_n$
for $t = 1$ to T **do**
  Receive instance $\boldsymbol{x}^t \in [0,1]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} +1 & \text{if } (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

  Receive label y^t and update

$$\boldsymbol{u}^{t+1} = \begin{cases} \boldsymbol{u}^t & \text{if } \hat{y}^t = y^t \\ \boldsymbol{u}^t + \eta\, y^t\, (\boldsymbol{u}^t \odot \boldsymbol{x}^t) & \text{otherwise} \end{cases}$$

Theorem 7 (Winnow bound [47, 71])**.** *Given any sequence of examples $(\boldsymbol{x}^t, y^t)$ such that $\boldsymbol{x}^t \in [0,1]^n$ the labels y^t are ± 1, and there is a weight $\boldsymbol{s}$ with k ones and $n - k$ zeros such that*

$$\boldsymbol{s} \cdot \boldsymbol{x}^t = \begin{cases} \geq \frac{1}{2} & \text{if } \; y^t = +1 \\ 0 & \text{otherwise}\,, \end{cases}$$

then the Winnow algorithm makes at most $7.18\, k \log \frac{n}{k}$ mistakes on this sequence, when $\eta \approx 1.28$, $\theta = 0.19$, and $w^1 = {}^n\!/_k$.

Note for the sake of simplicity we only address the case when there exists a consistent disjunction. In the more general case, there are the additional terms in the mistake bound that involve the number of attribute errors w.r.t best disjunction.

Theorem 8 (Reparameterized Winnow bound)**.** *Given any sequence of examples* $(\boldsymbol{x}^t, y^t)$ *and given that the assumptions of Theorem 7 hold, then the reparameterized Winnow algorithm makes at most* $5.66\, k \log \frac{n}{k}$ *mistakes on this sequence, when* $\eta \approx 0.85$*,* $\theta = 0.18$*, and* $u^1 = \sqrt{n/k}$*.*

Proof. We lower-bound the per trial progress $D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})$ towards any comparator $\boldsymbol{s}$ which satisfies the constraints in Theorem 7. Note that if no mistake occurs, $\boldsymbol{u}^{t+1} = \boldsymbol{u}^t$. Otherwise, $\boldsymbol{u}^{t+1}$ is updated to $\boldsymbol{u}^t + \eta\, y^t\, (\boldsymbol{u}^t \odot \boldsymbol{x}^t) = \boldsymbol{u}^t \odot (\boldsymbol{1} + \eta\, y^t\, \boldsymbol{x}^t)$. Assuming $\eta \leq 1$, and using the facts that $\log(1 + \eta y^t x_i^t) \geq x_i^t \log(1 + \eta y)$ and $(x_i^t)^2 \leq x_i^t$ for $x_i^t \in [0, 1]$, we have

$$\begin{aligned} D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) &= 2\boldsymbol{s} \cdot \log(\boldsymbol{1} + \eta y^t \boldsymbol{x}^t) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot (2\eta y^t \boldsymbol{x}^t + \eta^2 \boldsymbol{x}^t \odot \boldsymbol{x}^t) \\ &\geq 2\, \boldsymbol{s} \cdot \boldsymbol{x}^t\, \log(1 + \eta\, y^t) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\, (2\, \eta\, y^t + \eta^2)\,. \end{aligned}$$

A mistake occurs if $y^t \neq \hat{y}^t$. If $y^t = -1$, then $\boldsymbol{s} \cdot \boldsymbol{x}^t = 0$ by the assumption and $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \geq \theta$. Thus, the lower-bound on the progress becomes

$$(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t (2\, \eta - \eta^2) \geq \theta (2\, \eta - \eta^2)\,.$$

If $y^t = +1$, then we have $\boldsymbol{s} \cdot \boldsymbol{x}^t \geq \frac{1}{2}$ by the assumption and also $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \leq \theta$. Thus in this case the progress is lower-bounded by

$$\log(1 + \eta) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\, (2\, \eta\, y^t + \eta^2) \geq \log(1 + \eta) - \theta (2\, \eta + \eta^2)\,.$$

Setting the two values to be equal, we obtain $\theta = \frac{\log(1+\eta)}{4\, \eta}$. With this choice of θ,

the progress per mistake is always lower-bounded by $1/4\,(2-\eta)\,\log(1+\eta)$ which is maximized for $\eta \approx 0.85$. This choice of η yields $\theta \approx 0.18$ and the progress is ≈ 0.18. Summing over all trials and denoting the number of mistakes by M, we have

$$\underbrace{D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1)}_{k \log \frac{n}{k}} - \underbrace{D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{T+1} \odot \boldsymbol{u}^{T+1})}_{\geq 0} \geq 0.18\, M\,.$$

This implies $M \leq 5.56\, k\, \log \frac{n}{k}$. □

Note that the constant in the bound in Theorem 8 is slightly better than the one in Theorem 7. We believe this is merely a byproduct of the analysis and the two algorithms behave very closely in practice. A more thorough experimental as well as theoretical treatment is required to validate this argument.

4.4 Reparameterization of the Hedge

An important algorithm in the online expert setting is the Randomized Weighted Majority algorithm [46]. Here, we only discuss the simplified version known as the Hedge algorithm [29]. This algorithm maintains a non-negative probability vector $\boldsymbol{w}^t \in \Delta^{n-1}$ such that $\sum_i w_i^t = 1$. At trial t, the algorithm draws an expert/feature i with probability w_i and upon receiving the loss $\boldsymbol{\ell}^t$, it incurs the expected loss $\boldsymbol{w}^t \cdot \boldsymbol{\ell}^t$. The weights are then updated by a multiplicative exponentiated gradient term and re-normalized afterwards (see Algorithm 3) to assure that the non-negative weights sum to one. The Hedge update and its reparameterization are a special case on EG and reparameterized EG when the losses are the dot loss $\tilde{\boldsymbol{w}} \cdot \boldsymbol{\ell}^t$ and $\tilde{\boldsymbol{u}} \odot \tilde{\boldsymbol{u}} \cdot \boldsymbol{\ell}^t$, respectively.

The normalization ensures that $\boldsymbol{w}^{t+1} \in \Delta^{n-1}$. The Hedge update at round t is motivated by minimizing the relative entropy to the current weight $\boldsymbol{w}^t$ plus the

dot loss:

$$\boldsymbol{w}^{t+1} = \operatorname*{arg\,min}_{\tilde{\boldsymbol{w}} \in \Delta^{n-1}} \left\{ {}^{1}\!/_{\eta}\, D_{\mathrm{RE}}(\tilde{\boldsymbol{w}}, \boldsymbol{w}^t) + \tilde{\boldsymbol{w}} \cdot \boldsymbol{\ell}^t \right\}. \tag{4.10}$$

Algorithm 3 Hedge Algorithm

Parameters initial probability vector $\boldsymbol{w}^1 \in \Delta^{n-1}$, learning rate η

for $t = 1$ to T **do**

Draw expert i with probability w_i^t

Incur loss ℓ_i^t & expected loss $\boldsymbol{w}^t \cdot \boldsymbol{\ell}^t$

Update $\boldsymbol{w}^{t+1} = \dfrac{\boldsymbol{w}^t \odot \exp(-\eta\, \boldsymbol{\ell}^t)}{\sum_i \boldsymbol{w}_i^t \exp(-\eta\, \ell_i^t)}$

Algorithm 4 Reparameterized Hedge Alg.

Parameters initial weight vector $\boldsymbol{u}^1 \in \mathbb{R}^n$ s.t. $\|\boldsymbol{u}^1\|_2 = 1$, learning rate η

for $t = 1$ to T **do**

Draw expert i with probability $(u_i^t)^2$

Incur loss ℓ_i^t & expected loss $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t$

Update $\boldsymbol{u}^{t+1} = \dfrac{\boldsymbol{u}^t - \eta\, \boldsymbol{u}^t \odot \boldsymbol{\ell}^t}{\|\boldsymbol{u}^t - \eta\, \boldsymbol{u}^t \odot \boldsymbol{\ell}^t\|_2}$

Theorem 9 (Hedge bound [46, 29])**.** *For any sequence of loss vectors* $\{\boldsymbol{\ell}^t\}_{t=1}^T \in [0,1]^n$ *such that* $\|\boldsymbol{\ell}^t\|_\infty < L$*, any comparator* $\boldsymbol{s} \in \Delta^{n-1}$ *and any start vector* $\boldsymbol{w}^1 \in \Delta^{n-1}$ *such that* $D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) \le D \le \log n$*, the total expected loss of the Hedge algorithm with start vector* $\boldsymbol{w}^1$ *and learning rate* $\eta = \log(1 + \sqrt{2D/L})$ *is bounded as*

$$\sum_t \boldsymbol{w}^t \cdot \boldsymbol{\ell}^t \le \sum_t \boldsymbol{s} \cdot \boldsymbol{\ell}^t + \sqrt{2LD} + D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1)\,.$$

In the reparameterized Hedge update, $\boldsymbol{w}$ is replaced with $\boldsymbol{u} \odot \boldsymbol{u}$ where $\boldsymbol{u} \in \mathbb{R}^n$ and $\|\boldsymbol{u}\|_2 = 1$. That is, the reparameterized weight $\boldsymbol{u}$ lies on the unit sphere and the squared weight $\boldsymbol{u} \odot \boldsymbol{u}$ corresponds to an n-dimensional probability vector. The update is motivated by minimizing the squared Euclidean distance as the inertia

term plus the expected loss[3],

$$\boldsymbol{u}^{t+1} = \operatorname*{arg\,min}_{\tilde{\boldsymbol{u}} \text{ s.t. } \|\tilde{\boldsymbol{u}}\|_2=1} \left\{ 1/\eta \, \|\tilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + (\tilde{\boldsymbol{u}} \odot \tilde{\boldsymbol{u}}) \cdot \boldsymbol{\ell}^t \right\}. \tag{4.11}$$

Using a Lagrange multiplier to enforce the constraint that $\sum_i (u_i^t)^2 = 1$ and solving for $\boldsymbol{u}^{t+1}$ yields the reparameterized Hedge update in Algorithm 4.

Theorem 10 (Reparameterized Hedge bound)**.** *For any sequence of loss vectors* $\{\boldsymbol{\ell}^t\}_{t=1}^T \in [0,1]^n$ *such that* $\|\boldsymbol{\ell}^t\|_\infty < L$*, any comparator* $\boldsymbol{s} \in \Delta^{n-1}$ *and any start vector* $\boldsymbol{u}^1 \in \mathbb{R}^n$ *such that* $\|\boldsymbol{u}^1\|_2 = 1$ *and* $D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \leq D \leq \log n$*, the total expected loss of reparameterized Hedge with learning rate* $\eta = (1 + \sqrt{L/D})^{-1}$ *is bounded as*

$$\sum_t (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t \leq \sum_t \boldsymbol{s} \cdot \boldsymbol{\ell}^t + 2\sqrt{LD} + D_{RE}(\boldsymbol{s}, \boldsymbol{w}^1)\,.$$

Proof. We lower-bound the progress of the algorithm towards an arbitrary comparator $\boldsymbol{s} \in \Delta^{n-1}$. Assuming $\eta < 1$, the progress can be written as

$$\begin{aligned} &D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ &\qquad = 2\,\boldsymbol{s} \cdot \log\left(1 - \eta\,\boldsymbol{\ell}^t\right) - \log\left((\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot (\mathbf{1} - 2\eta\,\boldsymbol{\ell}^t + \eta^2\,\boldsymbol{\ell}^t \odot \boldsymbol{\ell}^t)\right) \\ &\qquad \geq 2\,\boldsymbol{s} \cdot \boldsymbol{\ell}^t \log(1-\eta) - \log\left(1 - (2\eta - \eta^2)\,(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t\right). \end{aligned}$$

Using $\log(1-x) \geq -x/(1-x)$ and $-\log(1-x) \geq x$ for $0 \leq x < 1$ yields

$$D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \geq -\frac{2\,\eta}{1-\eta}\,\boldsymbol{s} \cdot \boldsymbol{\ell}^t + (2\eta - \eta^2)\,(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t\,.$$

Summing over all trials and re-arranging the term results in the following bound

[3] A similar first-order Taylor approximation to (4.7) is required to obtain the explicit update.

on the loss of the algorithm

$$\sum_t (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t \leq \frac{\frac{2\,\eta}{1-\eta} \sum_t \boldsymbol{s} \cdot \boldsymbol{\ell}^t + D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) - D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{T+1} \odot \boldsymbol{u}^{T+1})}{2\,\eta - \eta^2}\,.$$

Setting η to $(1 + \sqrt{L/D})^{-1}$ and substituting the values of L and D yields the bound. □

Note that the regret bound of Theorem 10 for the reparameterized Hedge has an additional $\sqrt{2}$ factor before the square root term in its regret bound. By plotting the progress, we can show that this additional factor disappears if you use the alternate learning rate $\eta = \sqrt{D/(D + 2\,L)}$. Based on this evidence, we conjecture that with the alternate tuning, the reparameterized Hedge has the same regret bound as the original.

Another approximation of the Hedge update has been analysed in [18], called the *Prod* update. It replaces the exponential factors $\exp(-\eta\,\ell_i)$ used in Hedge by their Taylor expansions $(1 - \eta\,\ell_i)$ and normalizes multiplicatively.[4] Our reparameterized GD update is subtly different: it is a GD update w.r.t. the square roots of the weights as the parameters.

4.5 Reparameterizations of EGU and EG for Linear Regression

The regret bounds for linear regression using GD, EG, and EGU have been analyzed extensively in [40]. Here, we derive similar bounds for the reparameterized EG and EGU updates.

[4] The same Taylor expansion is used in the *Approximated EG* update of [40], but that update keeps the weight sum as one by subtracting a term. It is motivated by the χ^2-divergence.

We first recall the original EGU algorithm for linear regression,. The unnormalized EG (i.e. EGU) algorithm maintains a weight vector $\boldsymbol{w}^t \in \mathbb{R}^n_{\geq 0}$. Upon receiving input $\boldsymbol{x}^t \in \mathbb{R}^n$ at round t, the algorithm predicts with $\hat{y}^t = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$. Then, the algorithm receives the response y^t and incurs loss $\|y^t - \hat{y}^t\|^2$. EGU updates the weights $\boldsymbol{w}^{t+1}$ as

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t \odot \exp\Big(-2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t \Big)\,.$$

[40] analyze a slight variant of EGU. For a given sequence of trials $(\boldsymbol{x}^t, y^t)$ for which $y^t \in [0, Y]$ for all t and for some $Y > 0$, the EGU predicts with $\hat{y}^t = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$ if $\hat{y}^t \leq Y$ holds, and sets $\hat{y}^t = Y$ otherwise. The variant uses the clipped $\hat{y}^t$ in its update.

Theorem 11 (Linear regression with EGU [40])**.** *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequence such that $\boldsymbol{x}^t \in [0, X]^n$ and $y^t \in [0, Y]$ for some constants $X, Y > 0$. Then for any comparator $\boldsymbol{s} \in [0, \infty)^n$, EGU with learning rate $\eta = 1/(3XY)$ and arbitrary start point $\boldsymbol{w}^1 \in [0, \infty)^n$ satisfies the total loss bound*

$$\sum_{t=1}^{T} \|y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t\|_2^2 \leq 3\Big(\sum_t \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 + X\,Y\,D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) \Big)\,. \tag{4.12}$$

Furthermore, let L and D be constants such that $\sum_t \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 \leq L$ and $D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) \leq D$ and let

$$\eta = \frac{\sqrt{D}}{\sqrt{2LXY} + 2XY\sqrt{D}}\,. \tag{4.13}$$

Then we have

$$\sum_{t=1}^{T} \|y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t\|_2^2 \leq \sum_{t=1}^{T} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 + 2\sqrt{2LXY\,D} + 2\,XY\,D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1)\,. \tag{4.14}$$

Reparameterized EGU uses $\boldsymbol{u}^t \odot \boldsymbol{u}^t$ as its weights, where $\boldsymbol{u}^1 \in \mathbb{R}^n$, and updates as

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{u}^t \odot \boldsymbol{x}^t\,, \tag{4.15}$$

where $\hat{y}^t$ is again the clipped prediction $\min\left(\boldsymbol{w}^t \cdot \boldsymbol{x}^t, Y\right)$.

Theorem 12 (Reparameterized EGU linear regression)**.** *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequence such that $\boldsymbol{x}^t \in [0, X]^n$ and $y^t \in [0, Y]$ for some constants $X, Y > 0$. Then for any comparator $\boldsymbol{s} \in [0, \infty)^n$, Reparameterized EGU with learning rate $\eta = 1/(3XY)$ and arbitrary start point $\boldsymbol{w}^1 \in [0, \infty)^n$ satisfies the same total loss bound* (4.12) *as the original EGU when replacing $\boldsymbol{w}^t$ with $\boldsymbol{u}^t \odot \boldsymbol{u}^t$. Moreover, for the same constants L and D as in Theorem 11 and by setting η to* (4.13)*, the reparameterized EGU achieves the same bound as in* (4.14)*.*

Proof sketch. We first establish a lower-bound of the form

$$a\,\|y^t - \hat{y}^2\|^2 - b\,\|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|^2 \le D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})$$

on the progress of the algorithm towards the comparator $\boldsymbol{s}$, for some constants $0 \le a, b$. We can lower-bound the progress as

$$\begin{aligned} &D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ &\ge \frac{2\,\boldsymbol{s} \cdot \boldsymbol{x}^t\,\log(1 - \eta\,(\hat{y}^t - y^t)\,X)}{X} + \left(2\,\eta\,(\hat{y}^t - y^t) - \eta^2\,(\hat{y}^t - y^t)^2\,X\right)(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\,. \end{aligned}$$

Denoting by $r \doteq \boldsymbol{s} \cdot \boldsymbol{x}^t$ and $q \doteq (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t$, it suffices to show that $G(q, \hat{y}, y, r) \le 0$ where (omitting the superscript t)

$$G(q, \hat{y}, y, r) = -\frac{2r\log(1 - \eta\,(\hat{y} - y)X)}{X} - \left(2\,\eta\,(\hat{y} - y) - \eta^2\,(\hat{y} - y)^2\,X\right)q - \|y - \hat{y}\|^2 + b\,\|y - r\|^2\,.$$

It suffices to show the result for $q = \hat{y}$. The function $G(\hat{y}, r, y, \hat{y})$ is maximized for $r = y - \log(1 - \eta(\hat{y} - y)X)/(Xb)$. Setting $\eta = b/(1 + 2XYb)$, the inequality holds for any $a \leq b/(1 + 2XYb)$. Setting $b = 1/(XY)$, the bound is achieved for $\eta = a = 1/(3XY)$. □

Algorithm 5 EGU Linear Regression

Parameters initial weight $\boldsymbol{w}^1 \in \mathbb{R}^n_{\geq 0}$, learning rate η

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0, X]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} \boldsymbol{w}^t \cdot \boldsymbol{x}^t & \text{if } \boldsymbol{w}^t \cdot \boldsymbol{x}^t \leq Y \\ Y & \text{otherwise} \end{cases}$$

Receive label y^t and update

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t \odot \exp\Big(-2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t \Big)$$

Algorithm 6 Reparam. EGU Lin. Reg.

Parameters initial weight $\boldsymbol{u}^1 \in \mathbb{R}^n$, learning rate η

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0, X]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} \boldsymbol{w}^t \cdot \boldsymbol{x}^t & \text{if } \boldsymbol{w}^t \cdot \boldsymbol{x}^t \leq Y \\ Y & \text{otherwise} \end{cases}$$

where $\boldsymbol{w}^t = \boldsymbol{u}^t \odot \boldsymbol{u}^t$

Receive label y^t and update

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{u}^t \odot \boldsymbol{x}^t$$

Alternatively, the (normalized) EG algorithm maintains a probability vector $\boldsymbol{w}^t \in \Delta^{n-1}$ as its weight vector. The update for EG is the same as EGU, except for a multiplicative normalization:

$$\boldsymbol{w}^{t+1} = \frac{\boldsymbol{w}^t \odot \exp\Big(-2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t \Big)}{\sum_i w_i^{t+1} \exp\Big(-2\,\eta\,(\hat{y}^t - y^t)\,x_i^t \Big)}\,.$$

The following theorem from [40] expresses the worst-case bound of running EG for linear regression.

Theorem 13 (EG linear regression [40])**.** *Let* $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ *be any sequences such that* $\max_i x_i^t - \min_i x_i^t \leq R$ *for some* $R \geq 0$*. Then for any comparator* $\boldsymbol{s} \in \Delta^{n-1}$*, the EG algorithm with learning rate* $\eta = 2/(3R^2)$ *and arbitrary start point* $\boldsymbol{w}^1 \in$

Δ^{n-1} *satisfies the total loss bound*

$$\sum_{t=1}^{T} \|y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t\|_2^2 \leq \frac{3}{2} \Big(\sum_{t=1}^{T} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 + R^2 D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) \Big) . \tag{4.16}$$

Furthermore, let L *and* D *be constants such that* $\sum_t \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 \leq L$ *and* $D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) \leq D$ *and let*

$$\eta = \frac{2\sqrt{D}}{R\sqrt{2L} + R^2\sqrt{D}} . \tag{4.17}$$

Then we have

$$\sum_{t=1}^{T} \|y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t\|_2^2 \leq \sum_{t=1}^{T} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 + R\sqrt{2LD} + \frac{R^2}{2} D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{w}^1) . \tag{4.18}$$

Using (4.9), the reparameterized EG update for linear update can be written as

$$\boldsymbol{u}^{t+1} = \frac{\boldsymbol{u}^t - \eta \, (\hat{y}^t - y^t) \, \boldsymbol{u}^t \odot \boldsymbol{x}^t}{\|\boldsymbol{u}^t - \eta \, (\hat{y}^t - y^t) \, \boldsymbol{u}^t \odot \boldsymbol{x}^t\|_2} . \tag{4.19}$$

The algorithm for EG on linear regression is similar to Algorithm 6, except the initial weight $\boldsymbol{u}^1$ satisfies $\|\boldsymbol{u}^1\|_2 = 1$ and the update is replaced with (4.19).

In the following, we show the regret for the reparameterized EG update for linear regression. For simplicity, we first consider the case when the input $\boldsymbol{x} \in [0, X]^n$ for some $X > 0$.

Theorem 14 (Reparameterized EG linear regression)**.** *Let* $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ *be any sequences such that* $\boldsymbol{x}^t \in [0, X]^n$ *for some* $X \geq 0$*. Then for any comparator* $\boldsymbol{s} \in \Delta^{n-1}$*, the Reparameterized EG algorithm with learning rate* $\eta = 1/(3X^2)$ *and arbitrary start point* $\boldsymbol{u}^1 \in \mathbb{R}^n$*, such that* $\|\boldsymbol{u}^1 \odot \boldsymbol{u}^1\|_2 = 1$*, satisfies the total loss bound*

$$\sum_{t=1}^{T} \|y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\|_2^2 \leq 3 \Big(\sum_{t} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 + X^2 D_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \Big) . \tag{4.20}$$

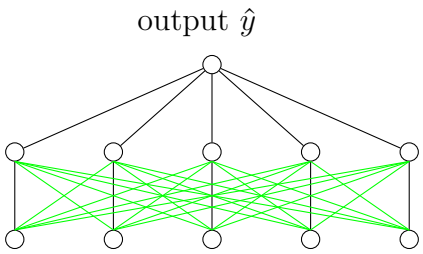


Figure 4.2: Complete two-layer linear network. The green weights are initialized to zero.

Furthermore, let L *and* D *be constants such that* $\sum_t \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 \leq L$ *and* $D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \leq D$ *and let*

$$\eta = \frac{\sqrt{D}}{X\sqrt{2L} + 2\,X^2\sqrt{D}}\,. \tag{4.21}$$

Then we have

$$\sum_{t=1}^{T} \|y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t)\cdot \boldsymbol{x}^t\|_2^2 \leq \sum_{t=1}^{T} \|y^t - \boldsymbol{s}\cdot\boldsymbol{x}^t\|_2^2 + 2\,X\sqrt{2\,LD} + 2\,X^2\,D_{\text{RE}}(\boldsymbol{s}, \boldsymbol{w}^1)\,. \tag{4.22}$$

The proof is given in the Appendix A. In the following, we claim a bound for the reparameterized EG for the more general case where $\boldsymbol{x}^t \in [-X, X]^n$ for all t. We provide a proof sketch for the claim in the appendix. Our partial proof shows strong evidence for the existence of such regret bound. However, finding an analytical proof by simplifying the final form is left for future work.

Claim 1 (A more general bound for reparameterized EG linear regression)**.** *For the same setting as in Theorem 14 but a more general case where* $\boldsymbol{x}^t \in [-X, X]^n$ *for all* t*, the reparameterized EG algorithm achieves the same bound* (4.22) *with* $R = 2X$ *when* η *is set to* (4.21)*.*

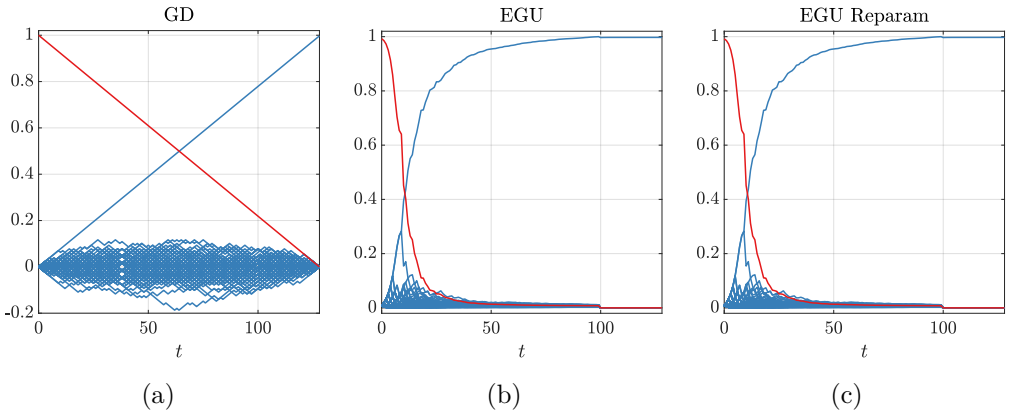


Figure 4.3: GD, EGU, & Reparameterized EGU on the online Hadamard problem ($n=128$): at round t, we train until consistency on the t past examples. The 128 weights are shown in blue and the average loss over all 128 examples in red.

4.6 Simulations

In this section, we provide a number of simulations in order to discuss the implications of the results of the previous sections. We first show some experimental results on the behavior of the GD as well as EGU and the reparameterized EGU updates for learning the Hadamard problem. Next, we show the effect of different initializations on the results.

4.6.1 Lower-bounds on the Hadamard Problem

Linear lower-bounds for all three cases have been studied in the previous work [39, 67]. These lower-bounds are all based on the "Hadamard problem"[5]: The instances are the rows of the n-dimensional Hadamard matrix in random order and the target is one of the columns. This means the target is a unit weight vector that "selects" the target column. GD brings up the weight of the right column slowly at the expense of the linear decaying average square loss. Multiplicative updates, such as EGU, bring up the target weight dramatically faster

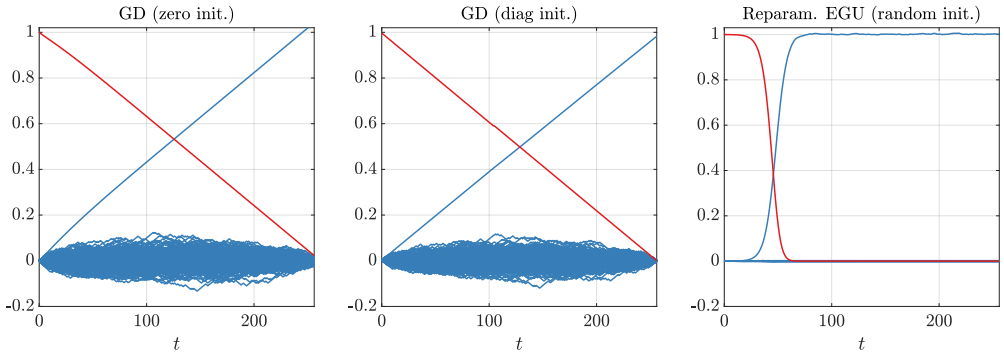
[5]Similar behavior is observed if a random ± 1 matrix is used.

and the average square loss decays essentially after $\log n$ examples (Figure 4.3). GD on the u_i of the sparse linear network of Figure 4.1 has the same behavior as EGU with almost identical weights trajectories. Surprisingly, if we run GD on the complete (i.e. fully-connected) two-layer network (Figure 4.2) with the green weights initialized to zero, then the combined linear weights of both layers again behave as when a simple linear neuron is trained with GD (see Section 4.6.2). It seems that GD focuses on keeping the highest weight as small as possible and uses the additional weights to overfit.

4.6.2 Behavior of GD and Reparameterized EGU with Different Initializations

Here, we discuss two interesting observations on the two-layer networks in Figure 4.1 and Figure 4.2. We extend the experiments on learning a column of the Hadamard matrix to two-layer networks. We consider the online Hadamard problem ($n = 128$) where we perform one pass over the examples. The results are shown in Figure 4.4. The figures stress two points about training two-layer linear networks with GD. First, if all the weights in the first layer are initialized to zero (Figure 4.4(a)), then the product of the weight matrix of the first layer times the weight vector of the second layer behaves qualitatively the same as the weights of a single linear neuron trained with GD (compare to Figure 4.3(a)). The same is true if the green weights in Figure 4.2 are initialized to zero and the remaining weights (black diagonal weights of the first layer and the weights of the second layer) are all set to uniform (Figure 4.4(b)). This is essentially the same initialization used for the reparameterized EGU algorithm, except in this case, the green connections are not removed.

Second, the initialization of the sparse linear network (Figure 4.1) is rather ro-



(a) (b) (c)

Figure 4.4: Results of two-layer linear networks on the online Hadamard problem ($n = 128$). We perform one pass over the examples. The product of the weights of the first and the second layer (128 weights in total) are shown in blue and the average loss over all 128 examples in red. Results of the two-layer network of Figure 4.2 using GD where the first layer weights are (a) initialized to zero, (b) initialized uniformly on both diagonals, (c) results of the sparse network of Figure 4.1 where the weights are initialized randomly.

bust. Equal initializations or random initializations of both diagonals all make the combined weight behave qualitatively like EGU on a single neuron (Figure 4.4(c) compared to Figure 4.3(b)).

4.7 Open problems

There are a number of technical open problems. A complete analytical proof still has to be provided for the regret of reparameterized EG when the instance domain is two-sided, i.e. $\boldsymbol{x}^t \in [-X, X]^n$. Proofs of such bounds are also required for the original and the reparameterized EGU. Experimentally, the updates behave as expected but obtaining analytical regret bounds (in particular for EGU) in those cases seems challenging. For the lower-bound discussion, we still need an analytical proof that the optimal initialization for the weights of the first layer in a two-layer linear network is always zero when the network is trained with GD

and the targets are columns of a Hadamard matrix or their negations. So far, we were only able to validate this by means of numerical differentiation.

Also, there are a number of open problems in the expert setting. We were able to repeat the regret bounds for the Reparameterized Hedge algorithm (i.e. when the loss is the dot loss and the loss components lie in $[0, 1]$). However, so far we were not able to repeat $\mathcal{O}(\log n)$ regret bounds [68, 33] for a reparameterized version of the multiplicative expert algorithm when the loss is the dot loss and the loss components are the square loss (a.k.a. the Brier scores). We believe that $\mathcal{O}(\log n)$ regret bounds are not possible for the reparameterized version of the expert algorithm when the loss components are log losses. The reason is that this loss is unbounded.

There are many open problems regarding neural networks: are reparameterized multiplicative updates useful for large networks and do they produce sparser weights?

Chapter 5

Tempered Bregman Divergence for Classification

5.1 Introduction

The logistic loss, also known as the softmax loss, has been the standard choice in training deep neural networks for classification. The loss involves the application of the softmax function on the activations of the last layer to form the class probabilities followed by the relative entropy (aka the Kullback-Leibler (KL) divergence) between the true labels and the predicted probabilities. The logistic loss is known to be a convex function of the activations (and consequently, the weights) of the last layer.

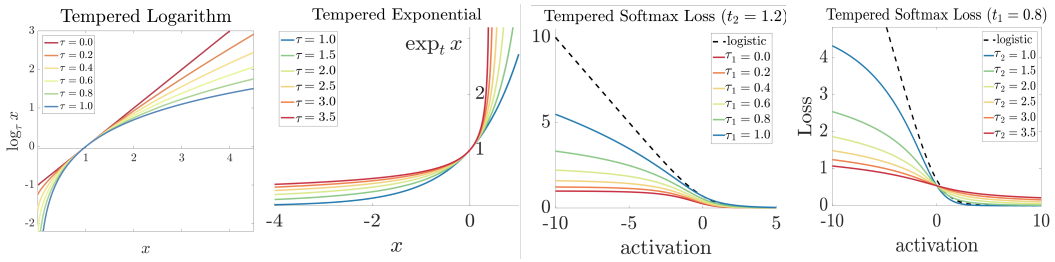
Although desirable from an optimization standpoint, convex losses have been shown to be prone to outliers [48] as the loss of each individual example unboundedly increases as a function of the activations. These outliers may correspond to extreme examples that lead to large gradients, or misclassified training examples that are located far away from the classification boundary. Requiring a convex loss function at the output layer thus seems somewhat arbitrary, in particular

since convexity in the last layer's activations does not guarantee convexity with respect to the parameters of the network outside the last layer. Another issue arises due to the exponentially decaying tail of the softmax function that assigns probabilities to the classes. In the presence of mislabeled training examples near the classification boundary, the short tail of the softmax probabilities enforces the classifier to stretch the decision boundary towards the noisy training examples. In contrast, heavy-tailed alternatives for the softmax probabilities have been shown to significantly improve the robustness of the loss to these examples [28].

The logistic loss is essentially the negative logarithm of the predicted class probabilities, which are computed as the normalized exponentials of the inputs. In this chapter, we tackle both shortcomings of the logistic loss, pertaining to its convexity as well as its tail-lightness, by replacing the logarithm and the exponential functions with their corresponding "tempered" versions. Recall that the tempered logarithm function $\log_\tau : \mathbb{R}_+ \to \mathbb{R}$ with *temperature* parameter $\tau \in \mathbb{R}$ as in [49] is defined as:

$$\log_\tau(x) \doteq \frac{1}{1-\tau}(x^{1-\tau} - 1)\,.$$

The $\log_\tau$ function is monotonically increasing and concave. The standard (natural) logarithm is recovered at the limit $\tau \to 1$. Unlike the standard log, the $\log_\tau$ function is bounded from below by $-1/(1-\tau)$ for $0 \leq \tau < 1$. This property will be used to define bounded loss functions that are significantly more robust to outliers. Similarly, our heavy-tailed alternative for the softmax function is based on the tempered exponential function. The function $\exp_\tau : \mathbb{R} \to \mathbb{R}_+$ with



(a) (b) (c) (d)

Figure 5.1: Tempered logarithm and exponential functions, and the bi-tempered logistic loss: (a) $\log_\tau$ function, (b) $\exp_\tau$ function, bi-tempered logistic loss when (c) $\tau_2 = 1.2$ fixed and $\tau_1 \leq 1$, and (d) $\tau_1 = 0.8$ fixed and $\tau_2 \geq 1$.

temperature $\tau \in \mathbb{R}$ is defined as the inverse[1] of $\log_\tau$, that is,

$$\exp_\tau(x) \doteq [1 + (1-\tau)\, x]_+^{1/(1-\tau)} , \tag{5.1}$$

where $[\,\cdot\,]_+ = \max\{\,\cdot\,, 0\}$. The standard exp function is again recovered at the limit $\tau \to 1$. Compared to the exp function, a heavier tail (for negative values of x) is achieved for $\tau > 1$. We use this property to define heavy-tailed analogues of softmax probabilities at the output layer.

The vanilla logistic loss can be viewed as a logarithmic (relative entropy) divergence that operates on a "matching" exponential (softmax) probability assignment [35, 38]. Its convexity then stems from classical convex duality, using the fact that the probability assignment function is the gradient of the dual function to the negative entropy on the simplex. When the $\log_{\tau_1}$ and $\exp_{\tau_2}$ are substituted instead, this duality still holds whenever $\tau_1 = \tau_2$, albeit with a different Bregman divergence, and the induced loss remains convex[2]. However, for $\tau_1 < \tau_2$, the loss becomes non-convex in the output activations. In particular, $0 \leq \tau_1 < 1$ leads to a bounded loss, while $\tau_2 > 1$ provides tail-heaviness. Figure 5.1 illustrates the tempered $\log_\tau$ and $\exp_\tau$ functions as well as examples of our proposed bi-

[1] When $0 \leq \tau < 1$, the domain of $\exp_\tau$ needs to be restricted to $-1/(1-\tau) \leq x$ for the inverse property to hold.

[2] In a restricted domain when $\tau_1 = \tau_2 < 1$, as discussed later.

tempered logistic loss function for a two-class problem expressed as a function of the activation of the first class. The true label is assumed to be class one.

Tempered generalizations of the logistic regression have been introduced before [27, 28, 75, 10]. The most recent two-temperature method [10] is based on the Tsallis divergence and contains all the previous methods as special cases. However, the Tsallis based divergences do not result in proper loss functions. In contrast, we show that the Bregman based construction introduced in this chapter is indeed proper, which is a requirement for many real-world applications. The material in chapter appeared as conference papers in AISTATS 2019 [10] and NeurIPS 2019 [5].

5.1.1 Our replacement of the softmax output layer in neural networks

Consider an arbitrary classification model with multiclass softmax output. We are given training examples of the form $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is a fixed dimensional input vector and the target $\boldsymbol{y}$ is a probability vector over k classes. In practice, the targets are often one-hot encoded binary vectors in k dimensions. Each input $\boldsymbol{x}$ is fed to the model, resulting in a vector $\boldsymbol{z}$ of inputs to the final softmax layer. This layer typically has one trainable weight vector $\boldsymbol{w}_i$ per class i and yields the predicted class probability

$$\hat{y}_i = \frac{\exp(\hat{a}_i)}{\sum_{j=1}^{k} \exp(\hat{a}_j)} = \exp\Big(\hat{a}_i - \log \sum_{j=1}^{k} \exp(\hat{a}_j)\Big),$$

for linear activation $\hat{a}_i = \boldsymbol{w}_i \cdot \boldsymbol{z}$ for class i. We first replace the softmax function by a generalized heavy-tailed version that uses the $\exp_{\tau_2}$ function with $\tau_2 > 1$,

which we call the *tempered softmax function*:

$$\hat{y}_i = \exp_{\tau_2}\Big(\hat{a}_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}})\Big), \quad \text{where} \quad \lambda_{\tau_2}(\hat{\boldsymbol{a}}) \in \mathbb{R} \;\; \text{is s.t.} \;\; \sum_{j=1}^{k} \exp_{\tau_2}\Big(\hat{a}_j - \lambda_{\tau_2}(\hat{\boldsymbol{a}})\Big) = 1\,.$$

This requires computing the normalization value $\lambda_{\tau_2}(\hat{\boldsymbol{a}})$ (for each example) via a binary search or an iterative procedure like the one given in Appendix B. The relative entropy between the true label $\boldsymbol{y}$ and prediction $\hat{\boldsymbol{y}}$ is replaced by the tempered version with temperature range $0 \le \tau_1 < 1$,

$$\sum_{i=1}^{k}\Big(y_i\left(\log_{\tau_1} y_i - \log_{\tau_1} \hat{y}_i\right) - \tfrac{1}{2-\tau_1}\left(y_i^{2-\tau_1} - \hat{y}_i^{2-\tau_1}\right)\Big)$$
$$\overset{\text{if } \boldsymbol{y} \text{ one-hot}}{=} \quad -\log_{\tau_1} \hat{y}_c - \tfrac{1}{2-\tau_1}\Big(1 - \sum_{i=1}^{k} \hat{y}_i^{2-\tau_1}\Big)\,.$$

where $c = \arg\max_i y_i$ is the index of the one-hot class. In later sections we prove various properties of this loss. When $\tau_1 = \tau_2 = 1$, then it reduces to the vanilla relative entropy loss with softmax. Also when $0 \le \tau_1 < 1$, then the loss is bounded, while $\tau_2 > 1$ gives the tempered softmax function a heavier tail.

5.1.2 An illustration

We provide some intuition on why both boundedness of the loss as well as tail-heaviness of the tempered softmax are crucial for robustness. For this, we train a small two-layer feed-forward neural network on a synthetic binary classification problem in two dimensions. The network has 10 and 5 units in the first and second layer, respectively[3]. Figure 5.2(a) shows the results of the logistic and our bi-tempered logistic loss on the noise-free dataset. The network converges to a desirable classification boundary (the white stripe in the figure) using both

[3]An interactive visualization of the bi-tempered loss is available at: `https://google.github.io/bi-tempered-loss/`

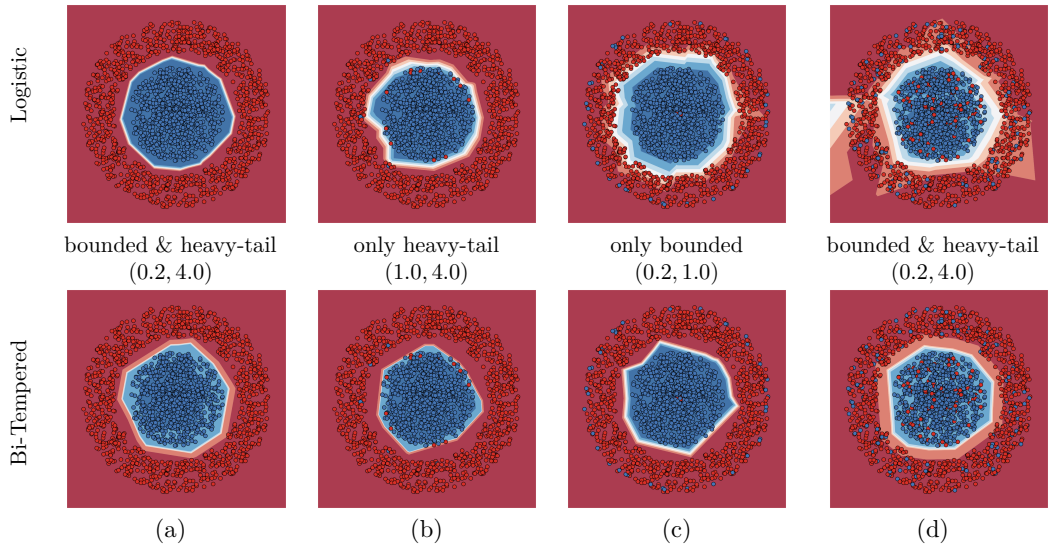


Figure 5.2: Logistic vs. robust bi-tempered logistic loss: (a) noise-free labels, (b) small-margin label noise, (c) large-margin label noise, and (d) random label noise. The temperature values (τ_1, τ_2) for the bi-tempered loss are shown above each figure.

loss functions. In Figure 5.2(b), we illustrate the effect of adding small-margin label noise to the training examples, targeting those examples that reside near the noise-free classification boundary. The logistic loss clearly follows the noisy examples by stretching the classification boundary. On the other hand, using *only* the tail-heavy tempered softmax function ($\tau_2 = 4$ while $\tau_1 = 1$, i.e. KL divergence as the divergence) can handle the noisy examples by producing more uniform class probabilities. Next, we show the effect of large-margin noisy examples in Figure 5.2(c), targeting examples that are located far away from the noise-free classification boundary. The convexity of the logistic loss causes the network to be highly affected by the noisy examples that are located far away from the boundary. In contrast, *only* the boundedness of the loss ($\tau_1 = 0.2$ while $\tau_2 = 1$, meaning that the outputs are vanilla softmax probabilities) reduces the effect of the outliers by allocating at most a finite amount of loss to each example. Finally, we show the effect of random label noise that includes both small-margin and large-margin

noisy examples in Figure 5.2(d). Clearly, the logistic loss fails to handle the noise, while our bi-tempered logistic loss successfully recovers the appropriate boundary. Note for random noise, we exploit *both* boundedness of the loss ($\tau_1 = 0.2 < 1$) as well as the tail-heaviness of the probability assignments ($\tau_2 = 4 > 1$).

The theoretical background as well as our treatment of the softmax layer of the neural networks are developed in later sections. In particular, we show that special discrete choices of the temperatures result in a large variety of divergences commonly used in machine learning. As we show in our experiments, tuning the two temperatures as continuous parameters is crucial.

5.2 Tempered Matching Loss

We start by discussing a few properties of the convex function F_τ and the corresponding tempered Bregman divergence. Note that the convexity of F_τ can be verified easily by considering the Hessian $\boldsymbol{H}_{F_\tau}(\boldsymbol{y}) = \nabla^2 F_\tau(\boldsymbol{y}) = \mathrm{diag}(\boldsymbol{y}^{-\odot\tau}) \succcurlyeq 0$ for any $\boldsymbol{y} \in \mathbb{R}^k_{\geq 0}$. We also showed the strong convexity of F_τ for $0 \leq \tau \leq 1$ in Chapter 3, Lemma 7. The following corollary is the direct consequence of the strong convexity of F_τ.

Corollary 8. *Let* $\max(\|\boldsymbol{y}\|_{2-\tau},\ \|\hat{\boldsymbol{y}}\|_{2-\tau}) \leq B$ *for* $0 \leq \tau < 1$. *Then*

$$\frac{1}{2B^\tau}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2_{2-\tau} \leq D_{F_\tau}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \leq \frac{B^\tau}{2\,(1-\tau)^2}\|\boldsymbol{y}^{1-\tau} - \hat{\boldsymbol{y}}^{1-\tau}\|^2_{\frac{2-\tau}{1-\tau}}\,.$$

Proof. Note that using the duality of the Bregman divergences, we have

$$D_{F_\tau}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = D_{F_t^*}(f_\tau(\hat{\boldsymbol{y}}),\ f_\tau(\boldsymbol{y})) = D_{F_t^*}(\log_\tau(\hat{\boldsymbol{y}}), \log_\tau(\boldsymbol{y}))\,.$$

Using the strong convexity of F_τ and strong smoothness of F_τ^*, we have

$$\frac{1}{2B^\tau}\,\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_{2-\tau}^2 \le D_{F_\tau}(\boldsymbol{y},\hat{\boldsymbol{y}}) \le \frac{B^\tau}{2}\,\|\log_\tau \boldsymbol{y}-\log_\tau \hat{\boldsymbol{y}}\|_{\frac{2-\tau}{1-\tau}}^2\,.$$

Note that $\|\cdot\|_{2-\tau}$ and $\|\cdot\|_{\frac{2-\tau}{1-\tau}}$ are dual norms. Substituting the definition of $\log_\tau$ to the right-hand-side, we have

$$\begin{aligned}\frac{B^\tau}{2}\|\log_\tau \boldsymbol{y}-\log_\tau \hat{\boldsymbol{y}}\|_{2-\tau}^2 = \frac{B^\tau}{2\,(1-\tau)^2}\,\|\boldsymbol{y}^{1-\tau}-\hat{\boldsymbol{y}}^{1-\tau}\|_{\frac{2-\tau}{1-\tau}}^2 \\ \le \frac{B^\tau}{2\,(1-\tau)^2}\left(2\,B^{1-\tau}\right)^2 = \frac{2\,B^{2-\tau}}{(1-\tau)^2}\,.\end{aligned}$$

□

Thus for $0 \le \tau < 1$, $D_{F_\tau}(\boldsymbol{y},\hat{\boldsymbol{y}})$ is upper-bounded by $\frac{2\,B^{2-\tau}}{(1-\tau)^2}$. Note that boundedness on the simplex also implies boundedness in the $L_{2-\tau}$-ball. Thus, Corollary 8 immediately implies the boundedness of the divergence $D_{F_\tau}(\boldsymbol{y},\hat{\boldsymbol{y}})$ with $0 \le \tau < 1$ over the simplex. Alternate parameterizations of the family $\{F_\tau\}$ of convex functions and their corresponding Bregman divergences are discussed in Appendix B.2.

5.2.1 Tempered softmax function

Now, let us consider the convex function $F_\tau(\boldsymbol{y})$ when its domain is restricted to the probability simplex Δ^{k-1}. We denote the constrained dual of $F_\tau(\boldsymbol{y})$ by $\check{F}_\tau^*(\boldsymbol{a})$,

$$\begin{aligned}\check{F}_\tau^*(\boldsymbol{a}) &= \sup_{\boldsymbol{y}'\in\Delta^{k-1}} \left(\boldsymbol{y}'\cdot\boldsymbol{a} - F_\tau(\boldsymbol{y}')\right) \\ &= \sup_{\boldsymbol{y}'\in\mathbb{R}^k_{\ge 0}} \inf_{\lambda_\tau\in\mathbb{R}} \left(\boldsymbol{y}'\cdot\boldsymbol{a} - F_\tau(\boldsymbol{y}') + \lambda_\tau\left(1-\sum_{i=1}^k y_i'\right)\right).\end{aligned} \tag{5.2}$$

Following our discussion in Chapter 2 and using (2.1), the transfer function induced by $\check{F}^*_\tau$ is[4]

$$\boldsymbol{y} = \exp_\tau\Big(\boldsymbol{a} - \lambda_\tau(\boldsymbol{a})\,\mathbf{1}\Big), \quad \text{with } \lambda_\tau(\boldsymbol{a}) \;\text{ s.t. } \quad \sum_{i=1}^{k} \exp_\tau\Big(a_i - \lambda_\tau(\boldsymbol{a})\Big) = 1. \tag{5.3}$$

5.2.2 Matching loss of tempered softmax

Finally, we derive the matching loss function L_{F_τ}. Plugging in (5.3) into (3.25), we have

$$L_t(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) = D_{F_\tau}\Big(\boldsymbol{y}, \exp_\tau(\hat{\boldsymbol{a}} - \lambda_\tau(\hat{\boldsymbol{a}})\,\mathbf{1})\Big).$$

Recall that by Proposition 2, this loss is convex in activations $\hat{\boldsymbol{a}} \in \mathbf{dom}(\check{f}^*) \cap \{\boldsymbol{a}' \in \mathbb{R}^k \mid \boldsymbol{a}' \cdot \mathbf{1} = 0\}$. In general, $\lambda_\tau(\boldsymbol{a})$ does not have a closed form solution. However, it can be easily approximated via an iterative method, e.g., a binary search. An alternative (fixed-point) algorithm for computing $\lambda_\tau(\boldsymbol{a})$ for $\tau > 1$ is given in Algorithm 7 in Appendix B.

5.3 Robust Bi-Tempered Logistic Loss

A more interesting class of loss functions can be obtained by introducing a "mismatch" between the temperature of the divergence function (3.25) and the temperature of the probability assignment function, i.e. the tempered soft-

[4]Note that due to the simplex constraint, the link function $\boldsymbol{y} = \check{f}^*_\tau(\boldsymbol{a}) = \nabla \check{F}^*_\tau(\boldsymbol{a}) = \exp_\tau\big(\boldsymbol{a} - \lambda_\tau(\boldsymbol{a})\,\mathbf{1}\big)$ is different from $f^{-1}_\tau(\boldsymbol{a}) = f^*_\tau(\boldsymbol{a}) = \nabla F^*_\tau(\boldsymbol{a}) = \exp_\tau(\boldsymbol{a})$, i.e., the gradient of the unconstrained dual.

max (5.3). That is, we consider loss functions of the following type:

$$\boxed{\begin{array}{c}\forall\, 0 \le \tau_1 < 1 < \tau_2 \colon L^{\tau_2}_{\tau_1}(\hat{\boldsymbol{a}} \mid \boldsymbol{y}) := D_{F_{\tau_1}}\Big(\boldsymbol{y}, \exp_{\tau_2}(\hat{\boldsymbol{a}} - \lambda_{\tau_2}(\hat{\boldsymbol{a}})\mathbf{1})\Big), \\ \text{with } \lambda_{\tau_2}(\hat{\boldsymbol{a}}) \text{ s.t. } \sum_{i=1}^{k} \exp_{\tau_2}\Big(a_i - \lambda_{\tau_2}(\boldsymbol{a})\Big) = 1.\end{array}} \tag{5.4}$$

We call this the *Bi-Tempered Logistic Loss*. As illustrated in our two-dimensional example in Section 5.1, both properties are crucial for handling noisy examples. The derivative of the bi-tempered loss is given in Appendix B.4. In the following, we discuss the properties of this loss for classification.

5.3.1 Properness and Monte-Carlo sampling

Let $P_{\text{UK}}(\boldsymbol{x}, y)$ denote the (unknown) joint probability distribution of the observed variable $\boldsymbol{x} \in \mathbb{R}^m$ and the class label $y \in [k]$. The goal of discriminative learning is to approximate the posterior distribution of the labels $P_{\text{UK}}(y \mid \boldsymbol{x})$ via a parametric model $P(y \mid \boldsymbol{x}; \Theta)$ parameterized by Θ. Thus the model fitting can be expressed as minimizing the following expected loss between the data and the model's label probabilities

$$\mathbb{E}_{P_{\text{UK}}(\boldsymbol{x})}\Big[D\Big(P_{\text{UK}}(y \mid \boldsymbol{x}), P(y \mid \boldsymbol{x}; \Theta)\Big)\Big], \tag{5.5}$$

where $D\Big(P_{\text{UK}}(y \mid \boldsymbol{x}), P(y \mid \boldsymbol{x}; \Theta)\Big)$ is any divergence measure between $P_{\text{UK}}(y \mid \boldsymbol{x})$ and $P(y \mid \boldsymbol{x}; \Theta)$. We use $D := D_{F_{t_1}}$ as the divergence and $P(i \mid \boldsymbol{x}; \Theta) \doteq P(y = i \mid \boldsymbol{x}; \Theta) = \exp_{\tau_2}(\hat{a}_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}}))$, where $\hat{\boldsymbol{a}}$ is the activation vector of the last layer given input $\boldsymbol{x}$ and Θ is the set of all weights of the network. Ignoring the constant

terms w.r.t. Θ, our loss (5.5) becomes

$$\mathbb{E}_{P_{\text{UK}}(\boldsymbol{x})}\Big[\sum_i \Big(- P_{\text{UK}}(i \mid \boldsymbol{x}) \log_\tau P(i \mid \boldsymbol{x}; \Theta) + \frac{1}{2-\tau} P(i \mid \boldsymbol{x}; \Theta)^{2-\tau}\Big)\Big] \tag{5.6a}$$

$$= -\mathbb{E}_{P_{\text{UK}}(\boldsymbol{x},y)}\Big[\log_\tau P(y \mid \boldsymbol{x}; \Theta)\Big] + \mathbb{E}_{P_{\text{UK}}(\boldsymbol{x})}\Big[\frac{1}{2-\tau}\sum_i P(i \mid \boldsymbol{x}; \Theta)^{2-\tau}\Big)\Big] \tag{5.6b}$$

$$\approx \frac{1}{N}\sum_n \Big(- \log_\tau P(y_n \mid \boldsymbol{x}_n; \Theta) + \frac{1}{2-\tau}\sum_i P(i \mid \boldsymbol{x}_n; \Theta)^{2-\tau}\Big)\,, \tag{5.6c}$$

where from (5.6b) to (5.6c), we perform a Monte-Carlo approximation of the expectation w.r.t. $P_{\text{UK}}(\boldsymbol{x}, y)$ using samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$. Thus, (5.6c) is an unbiased approximate of the expected loss (5.5), thus is a *proper* loss [72].

Following the same approximation steps for the Tsallis divergence used in [10], we have

$$\mathbb{E}_{P_{\text{UK}}(\boldsymbol{x})}\Big[\underbrace{-\sum_i P_{\text{UK}}(i \mid \boldsymbol{x}) \log_\tau \frac{P(i \mid \boldsymbol{x}; \Theta)}{P_{\text{UK}}(i \mid \boldsymbol{x})}}_{D_\tau^{\text{Tsallis}}\left(P_{\text{UK}}(y|\boldsymbol{x}),\, P(y|\boldsymbol{x};\Theta)\right)}\Big] \approx -\frac{1}{N}\sum_n \log_\tau \frac{P(y_n \mid \boldsymbol{x}_n; \Theta)}{P_{\text{UK}}(y_n \mid \boldsymbol{x}_n)}\,,$$

which, due to the fact that $\log_\tau \frac{a}{b} \neq \log_\tau a - \log_\tau b$ in general, requires access to the (unknown) conditional distribution $P_{\text{UK}}(y \mid \boldsymbol{x})$. In this case the approximation $-\frac{1}{N}\sum_n \log_\tau P(y_n \mid \boldsymbol{x}_n; \Theta)$ proposed in [10] by setting $P_{\text{UK}}(y_n \mid \boldsymbol{x}_n)$ to 1 is not an unbiased estimator of (5.5) and therefore, not proper.

5.3.2 Bayes-risk consistency

Another important property of a multiclass loss is the Bayes-risk consistency [61]. The conditional risk of the multiclass loss $\boldsymbol{l}(\hat{\boldsymbol{a}})$ with $l_i \doteq \ell(\hat{\boldsymbol{a}} | \, y = i)$, $i \in [k]$ is defined as

$$R(\boldsymbol{\eta}, \boldsymbol{l}(\hat{\boldsymbol{a}})) = \sum_i \eta_i \, l_i\,,$$

where $\eta_i \doteq P_{\text{UK}}(y = i|\, \boldsymbol{x})$.

Definition 4 (Bayes-risk Consistency)**.** *A Bayes-risk consistent loss for multi-class classification is the class of loss functions ℓ for which $\hat{\boldsymbol{a}}^\star$, the minimizer of $R(\boldsymbol{\eta}, \boldsymbol{l}(\hat{\boldsymbol{a}}))$, satisfies*

$$\arg\min_i \ell(\hat{\boldsymbol{a}}^\star|\, y = i) \subseteq \arg\max_i \eta_i\,.$$

Bayes-risk consistency of the two-temperature logistic loss based on the Tsallis divergence was shown in [10]. As expected, the tempered Bregman loss (5.4) is also Bayes-risk consistent even in the non-convex case.

Proposition 8. *The multiclass bi-tempered logistic loss $L^{\tau_2}_{\tau_1}(\hat{\boldsymbol{a}}\,|\,y)$ is Bayes-risk consistent.*

Proof. For the bi-tempered loss, we have

$$l_i = -\log_{\tau_1} \exp_{\tau_2}(\hat{a}_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}})) + \frac{1}{2-\tau_1} \sum_j \exp_{\tau_2}(\hat{a}_j - \lambda_{\tau_2}(\hat{\boldsymbol{a}}))^{2-\tau_1}\,.$$

Note that the second term is repeated for all classes $i \in [k]$. Also,

$$R(\boldsymbol{\eta}, \boldsymbol{l}(\hat{\boldsymbol{a}})) = -\sum_i \eta_i \log_{\tau_1} \exp_{\tau_2}(\hat{a}_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}})) + \frac{1}{2-\tau_1} \sum_i \exp_{\tau_2}(\hat{a}_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}}))^{2-\tau_1}\,.$$

The minimizer of $R(\boldsymbol{\eta}, \boldsymbol{l}(\hat{\boldsymbol{a}}))$ satisfies

$$\eta_i = \exp_{\tau_2}(\hat{a}^\star_i - \lambda_{\tau_2}(\hat{\boldsymbol{a}}^\star))\,.$$

Since $-\log_{\tau_1}$ is a monotonically decreasing function for $0 \leq \tau_1 < 1$, we have

$$\arg\min_i \ell(\hat{\boldsymbol{a}}^\star | \, y = i) = \arg\min_i -\log_{\tau_1} \exp_{\tau_2}(\hat{a}_i^\star - \lambda_{\tau_2}(\hat{\boldsymbol{a}}^\star)) \\ = \arg\max_i \hat{a}_i^\star \subseteq \arg\max_i \eta_i \, .$$

□

5.4 Experiments

We demonstrate the practical utility of the bi-tempered logistic loss function on a wide variety of image classification tasks. For moderate-size experiments, we use MNIST dataset of handwritten digits [44] and CIFAR-100, which contains real-world images from 100 different classes [42]. We use ImageNet-2012 [24] for large scale image classification, having 1000 classes. All experiments are carried out using the TensorFlow [1] framework. We use P100 GPU's for small-scale experiments and Cloud TPU-v2 for larger scale ImageNet-2012 experiments. An implementation of the bi-tempered logistic loss is available online at: `https://github.com/google/bi-tempered-loss`.

5.4.1 Corrupted labels experiments

For our moderate size datasets, i.e. MNIST and CIFAR-100, we introduce noise by artificially corrupting a fraction of the labels and producing a new set of labels for each noise level. For all experiments, we compare our bi-tempered loss function against the logistic loss.

For MNIST, we use a CNN with two convolutional layers of size 32 and 64 with a mask size of 5, followed by two fully-connected layers of size 1024 and 10. We apply max-pooling after each convolutional layer with a window size equal to 2 and use dropout during training with keep probability equal to 0.75. We use

Dataset	Loss	Label Noise Level					
		0.0	0.1	0.2	0.3	0.4	0.5
MNIST	Logistic	**99.40**	98.96	98.70	98.50	97.64	96.13
	Bi-Tempered $(0.5, 4.0)$	99.24	**99.13**	**99.02**	**98.62**	**98.56**	**97.69**
CIFAR-100	Logistic	74.03	69.94	66.39	63.00	53.17	52.96
	Bi-Tempered (0.8, 1.2)	**75.30**	**73.30**	**70.69**	**67.45**	**62.55**	**57.80**

Table 5.1: Top-1 accuracy on a clean test set for MNIST and CIFAR-100 datasets where a fraction of the training labels are corrupted.

Model	Logistic	Bi-tempered (0.9,1.05)
Resnet18	71.333 ± 0.069	$\mathbf{71.618} \pm 0.163$
Resnet50	76.332 ± 0.105	$\mathbf{76.748} \pm 0.164$

Table 5.2: Top-1 accuracy on ImageNet-2012 with Resnet18 and 50 architectures.

the AdaDelta optimizer [74] with 500 epochs and batch size of 128 for training. For CIFAR-100, we use a Resnet56 [34] model without batch norm from [32] with SGD + momentum optimizer trained for 50k steps with batch size of 128 and use the standard learning rate stair case decay schedule. For both experiments, we report the test accuracy of the checkpoint which yields the highest accuracy on an identically label-noise corrupted validation set. We search over a set of learning rates for each experiment. For both experiments, we exhaustively search over a number of temperatures within the range $[0.5, 1)$ and $(1.0, 4.0]$ for τ_1 and τ_2, respectively. The results are presented in Table 5.1 where we report the top-1 accuracy on a clean test set. As can be seen, the bi-tempered loss outperforms the logistic loss for all noise levels (including the noise-free case for CIFAR-100). Using our bi-tempered loss function the model is able to continue to perform well even for high levels of label noise whereas the accuracy of the logistic loss drops immediately with a much smaller level of noise.

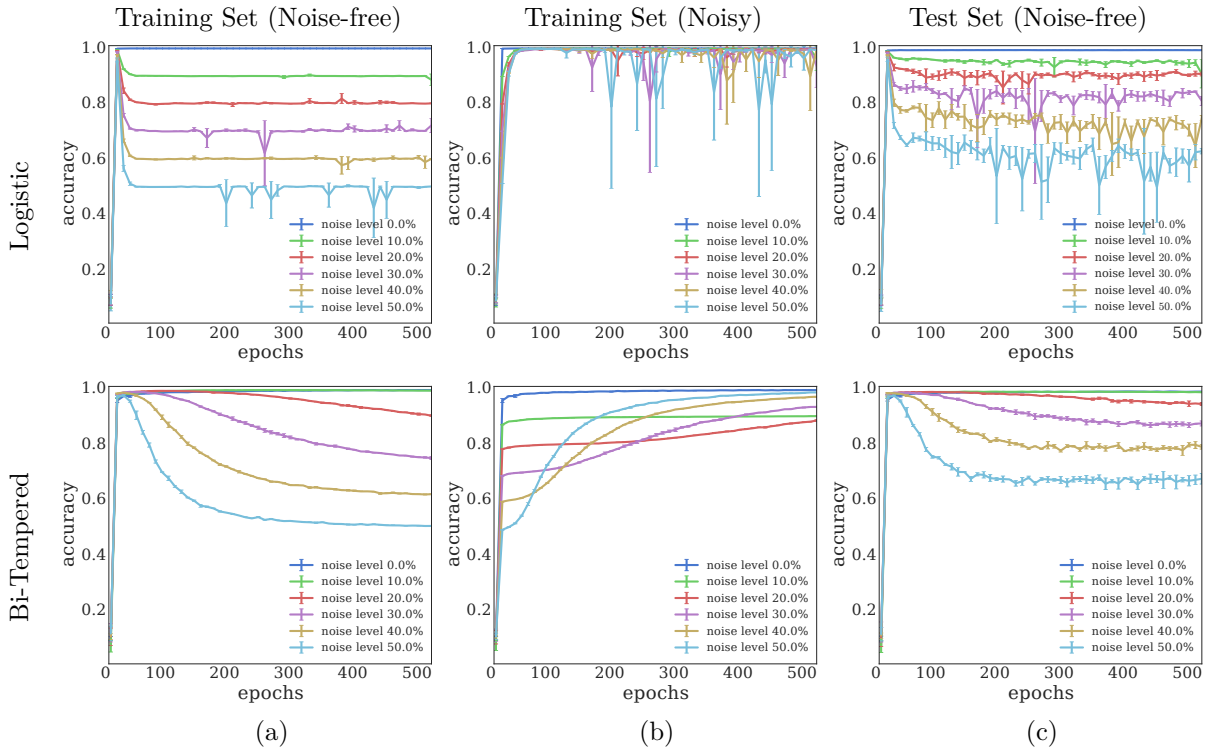


Figure 5.3: Top-1 accuracy of the models trained using the logistic loss (top) and the bi-tempered loss with $(\tau_1, \tau_2) = (0.5, 4.0)$ (bottom) on the noisy MNIST dataset: accuracy on (a) noise-free training set, (b) noisy training set, (c) and noise-free test set. Initially, both models provide better generalization but gradually overfit to the label noise. However, the overfitting for the logistic loss happens much earlier during the optimization. The variance of the model is also much higher for the logistic loss. The bi-tempered loss provides better generalization accuracy overall.

5.4.2 Overfitting to Noise and Generalization

We provide additional results on the generalization performance of the bi-tempered loss on the noisy MNIST dataset. In Figure 5.3, we illustrate the top-1 accuracy on the noise-free and noisy training set, as well the accuracy on the (noise-free) test set for both losses as a function of number of epochs. As can be seen from the figure, initially both models yield a relatively higher test accuracy, but gradually overfit to the label noise in the training set over time. The overfitting to the noise deteriorates the generalization capacity of the models. However,

overfitting to the noise happens earlier in the training and is much severe in case of the logistic loss. As a result, the final test accuracy (after 500 epochs) is comparatively much lower than the bi-tempered loss as the noise level increases. Finally, note that the variance of the model is also considerably higher for the logistic loss. This confirms that the bi-tempered loss results in more stable models when the data is noise-corrupted.

5.4.3 Large scale experiments

We train state-of-the-art Resnet18 and Resnet50 models on the ImageNet-2012 dataset. Note that the ImageNet-2012 dataset is inherently noisy due to some amount of mislabeling. We train on a 4x4 CloudTPU-v2 device with a batch size of 4096. All experiments were trained for 180 epochs, and use the SGD + momentum optimizer with staircase learning rate decay schedule. The results are presented in Table 5.2. For both architectures we see a significant gain in the top-1 accuracy using the robust bi-tempered loss.

5.5 Conclusion and Future Work

Neural networks on large standard datasets have been optimized along with a large variety of variables such as architecture, transfer function, choice of optimizer, and label smoothing to name just a few. We proposed a new variant by training the network with tunable loss functions. We do this by first developing convex loss functions based on temperature dependent logarithm and exponential functions. When both temperatures are the same, then a construction based on the notion of "matching loss" leads to loss functions that are convex in the last layer. However by letting the temperature of the new tempered softmax function be larger than the temperature of the tempered log function used in the divergence,

we construct tunable losses that are non-convex in the last layer. Our construction remedies two issues simultaneously: we construct bounded tempered loss functions that can handle large-margin outliers and introduce heavy-tailedness in our new tempered softmax function that seems to handle small-margin mislabeled examples. At this point, we simply took a number of benchmark datasets and networks for these datasets that have been heavily optimized for the logistic loss paired with vanilla softmax and simply replaced the loss in the last layer by our new construction. By simply trying a number of temperature pairs, we already achieved significant improvements. We believe that with a systematic "joint optimization" of all commonly tried variables, significant further improvements can be achieved. This is of course a more long-term goal. We also plan to explore the idea of annealing the temperature parameters over the training process.

Chapter 6

Conclusion and Future Work

In this thesis, we considered the application of the tempered Bregman divergence for developing continuous and discrete-time MD algorithms as well as robust classification loss functions. For this, we first developed a novel motivation for the continuous-time MD as the minimizer of a Bregman momentum plus loss. Next, we showed a reparameterization theorem for casting one MD update in terms of an another MD. Using the reparameterization theorem, we proved that the updates induced by the tempered Bregman divergence can be reparameterized as GD updates. As an example, we showed the application of the tempered updates for the linear regression problem. Finally, after giving an alternate motivation for the idea of a matching loss in terms of convex duality, we introduced the bi-tempered logistic loss for classification based on the tempered Bregman divergence. We showed that when the temperature of the softmax function matches the temperature of the Bregman divergence, we still obtain a convex loss function in activations. We also showed that introducing a certain mismatch between the temperatures results in non-convex robust losses for classification.

Apart from the open problems that we presented throughout the chapters, the following ideas can be explored as future research directions:

- The reparameterization trick introduced in Chapter 3 is more versatile and can be extended to other MD updates. An interesting extension of the reparameterization would be to cast other updates such as the p-norm perceptron [41] as GD updates. Also, it would be interesting to analyze the regret bounds for the reparameterized versions of such algorithms in the discrete case.

- A prominent research direction is to analyze the effect of the structure of a neural network on its learning dynamics. For instance, an open problem is to verify whether a multi-layer neural network can learn sparse solutions only via sparse structures. A potential impact of "learning the structure" is to impose certain regularizations naturally, without any explicit tuning.

- Another interesting direction is to consider other Bregman divergences (or possibly other class of divergences) to develop robust loss functions for regression, classification, and ranking. For ranking, an important notion is asymmetry of the loss to punish misplacement of the top-rank items more strictly than the remaining items. This can possibly be achieved using ideas similar to the bi-tempered loss, but instead by shifting and scaling the transfer function.

Appendix A

A.1 Proof of Theorem 12

Proof. We first establish a lower-bound of the form,

$$a\,\|y^t - \hat{y}^2\|_2^2 - b\,\|y^t - \boldsymbol{s}\cdot\boldsymbol{x}^t\|_2^2 \leq \Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})\,, \quad \text{(A.1)}$$

on the progress of the algorithm towards the comparator $\boldsymbol{s}$, for some constants $0 \leq a, b$. Assuming that $\eta \leq 1/(2XY)$, we have $(1 - \eta\,(\hat{y}^t - y^t)\,x_i^t) \geq 0$ for all i. Thus, we can lower-bound the progress as

$$\begin{aligned}
&\Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\
&\quad = 2\,\boldsymbol{s}\cdot\log(\mathbf{1} - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t) \\
&\quad - \log\left(\boldsymbol{w}^t\cdot(\mathbf{1} - 2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t + \eta\,(\hat{y}^t - y^t)^2\,\boldsymbol{x}^t \odot \boldsymbol{x}^t - \mathbf{1}\right).
\end{aligned}$$

Applying the inequalities $\log(1-\eta\,(\hat{y}^t-y^t)x_i^t) \geq \frac{x_i^t \log(1-\eta\,(\hat{y}^t-y^t)X)}{X}$ and $\log(1+x) \leq x$ for $x > -1$ and using the fact that $(x_i^t)^2 \leq x_i^t\,X$, we have

$$\begin{aligned}
&\Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\mathrm{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\
&\qquad \geq \frac{2\,\boldsymbol{s}\cdot\boldsymbol{x}^t\,\log(1 - \eta\,(\hat{y}^t - y^t)\,X)}{X} + \left(2\,\eta\,(\hat{y}^t - y^t) - \eta^2\,(\hat{y}^t - y^t)^2\,X\right)(\boldsymbol{u}^t \odot \boldsymbol{u}^t)\cdot\boldsymbol{x}^t\,.
\end{aligned}$$

Denoting by $r \doteq \boldsymbol{s}\cdot\boldsymbol{x}^t$ and $q \doteq (\boldsymbol{u}^t \odot \boldsymbol{u}^t)\cdot\boldsymbol{x}^t$, it suffices to show that $G(q,\hat{y},y,r) \leq 0$ where (omitting the superscript t)

$$G(q,\hat{y},y,r) = -\frac{2r\log(1-\eta\,(\hat{y}-y)X)}{X} - \Big(2\,\eta\,(\hat{y}-y) - \eta^2\,(\hat{y}-y)^2\,X\Big)\,q + a\,\|y-\hat{y}\|^2 - b\,\|y-r\|^2\,.$$

Recall that the prediction $\hat{y}^t$ of the reparameterized EGU is given by $\hat{y}^t = q = (\boldsymbol{u}^t \odot \boldsymbol{u}^t)\cdot\boldsymbol{x}^t$ if $q \leq Y$ holds; otherwise $\hat{y}^t = Y$. Thus, we need to show $G(q,\hat{y},y,r) \leq 0$ for two cases: when $\hat{y} = q$, and for $0 \leq \hat{y} = Y < q$. Recall that by the assumption $0 \leq y \leq Y$ and $\eta \leq 1/(2XY)$. Therefore, $G(q,\hat{y},y,r)$ is non-increasing in q for $\hat{y} \geq y$. Hence, the condition $G(q,\hat{y},y,r) \leq 0$ for $\hat{y} = Y < q$ is satisfied if $G(Y,Y,y,r) \leq 0$ holds. Thus, it suffices to show the result for $0 \leq \hat{y} = q \leq Y$.

For fixed $\hat{y}$ and y, the function $G(\hat{y},\hat{y},y,r)$ is maximized for

$$r = y - A/(Xb)\,,\quad \text{where} \quad A \doteq \log(1-\eta(\hat{y}-y)X)\,.$$

Plugging in this value, we have $H(\hat{y},y) \doteq G(\hat{y},\hat{y},y,y-A/(Xb))$ where

$$H(\hat{y},y) = -2\,\eta\,(\hat{y}-y)\hat{y} + \eta^2\,(\hat{y}-y)^2\,\hat{y}\,X + a\,(\hat{y}-y)^2 + A^2/(bX^2) - 2\,A\,y/X\,.$$

In order to obtain the bound (A.1), we show that $H(\hat{y},y) \leq 0$ holds for the choice of $\eta = b/(1+2XYb)$ and $a = b/(1+2XYb)$. Substituting these values for η and a, we have $H(y,y) = \frac{\partial H(y,y)}{\partial y} = 0$ and Furthermore,

$$\frac{\partial^2 H(\hat{y},y)}{\partial y^2} = -\frac{4b^2X(Y-\hat{y})}{(1+2XYb)^2} \leq 0 \quad \text{for all} \quad 0 \leq \hat{y} \leq Y\,.$$

Thus, it suffices to show that $\hat{y} = y$ is the only maximum of the function. For

this, we first write

$$\frac{\partial^2 H(\hat{y},y)}{\partial y^2} = -\frac{1}{\left(\frac{1}{2}+X\Big(Y+\frac{y}{2}-\frac{\hat{y}}{2}\Big)b\right)^2\left(\frac{1}{2}+XYb\right)^2} \times\left(\left(\frac{1}{2}+XYb\right)^2\log\Big(1+\frac{X(y-\hat{y})b}{1+2XYb}\Big)b + \Big(Y^3-\big(\frac{\hat{y}+y}{2}\big)Y^2+\big(\frac{\hat{y}^2-y^2}{4}\big)Y+\frac{\hat{y}(\hat{y}-y)}{8}\Big)b^4X^3 + \Big(Y^2-\big(\frac{\hat{y}+y}{2}\big)Y+\frac{\hat{y}^2-y^2}{8}\Big)b^3X^2+\Big(\frac{Y}{4}+\frac{\hat{y}-y}{8}\Big)b^2X\right).$$

Using the inequality $\log(1+x)\leq x$ for $x>-1$, we can upper-bound the log-term as

$$\log\Big(1+\frac{X(y-\hat{y})b}{1+2XYb}\Big)b \leq \frac{X(y-\hat{y})b^2}{1+2XYb},$$

and write the new function as $Q(\hat{y},y)$ such that $\frac{\partial^2 H(\hat{y},y)}{\partial y^2}\leq Q(\hat{y},y)$. It is trivial to check that the derivative

$$\frac{\partial Q(\hat{y},y)}{\partial y} = \frac{3\left(\Big(Y+\frac{y}{6}-\frac{\hat{y}}{2}\Big)\Big(Y+\frac{y}{2}-\frac{\hat{y}}{2}\Big)bX+Y+\frac{y}{3}-\frac{\hat{y}}{2}\right)X^2b^3}{2\left(\frac{1}{2}+X\Big(Y+\frac{y}{2}-\frac{\hat{y}}{2}\Big)b\right)^4}\geq 0,$$

for any $0\leq y\leq Y$ and $0\leq\hat{y}\leq Y$. Hence, it remains to check that $Q(\hat{y},Y)\leq 0$ holds:

$$Q(\hat{y},Y) = -\frac{1}{9\left(\frac{1}{3}+\Big(Y-\frac{\hat{y}}{3}\Big)bX\right)^3\left(\frac{1}{2}+bXY\right)^2}\times (Y-\hat{y})Xb^2\Big(\frac{1}{3}+\Big(Y-\frac{\hat{y}}{3}\Big)(Y-\hat{y})\Big(Y-\frac{\hat{y}}{2}\Big)b^3X^3+\frac{5\Big(Y^2-\frac{3}{5}Y\hat{y}+\frac{2}{15}\hat{y}^2\Big)b^2X^2}{2} + \frac{5\Big(Y-\frac{\hat{y}}{5}\Big)bX}{3}\Big)\leq 0,$$

which holds for any $0 \leq \hat{y} \leq Y$. This implies $\frac{\partial^2 H(\hat{y},y)}{\partial y^2} \leq Q(\hat{y},y) \leq 0$ for any $0 \leq \hat{y} \leq Y$ and $0 \leq y \leq Y$.

The remainder of the proof follows similarly to [40]. Specifically, by setting setting $b = c/(XY)$, we obtain

$$\sum_{t=1}^{T} \|y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\|^2 \leq \Big(1+2\,c\Big) \sum_{t=1}^{T} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|^2 + \Big(2+\frac{1}{c}\Big) XY \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1)\,.$$

Setting $c = 1$, the bound in (4.12) is achieved for $\eta = a = 1/(3XY)$. Using the values L and D and tuning for c achieves (4.14) for the choice of η as in (4.13).

□

A.2 Proof of Theorem 14

Similar to the proof of Theorem 12, we establish a lower-bound of the form

$$a\,\|y^t - \hat{y}^2\|_2^2 - b\,\|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|_2^2 \leq \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})\,, \quad \text{(A.2)}$$

on the progress of the algorithm towards the comparator $\boldsymbol{s}$ where $\sum_i u_i = 1$, and for some constants $0 \leq a, b$. Assuming that $\eta \leq 1/(2X^2)$, we have $(1 - \eta\,(\hat{y}^t - y^t)\,x_i^t) \geq 0$ for all i. Thus, we can lower-bound the progress as

$$\begin{aligned}
&\Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\
&\quad = 2\,\boldsymbol{s} \cdot \log(\mathbf{1} - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t) \\
&\quad - \log\Big(\boldsymbol{w}^t \cdot (\mathbf{1} - 2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t + \eta\,(\hat{y}^t - y^t)^2\,\boldsymbol{x}^t \odot \boldsymbol{x}^t\Big)\,.
\end{aligned}$$

Applying the inequalities $\log(1-\eta\,(\hat{y}^t-y^t)x_i^t) \geq \frac{x_i^t \log(1-\eta\,(\hat{y}^t-y^t)X)}{X}$ and $-\log(1-c) \geq c$ for $0 \leq c \leq 1$ and using the fact that $(x_i^t)^2 \leq x_i^t\, X$, we have

$$\begin{aligned}\Delta_{\text{RE}}(\boldsymbol{s},\, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ \geq \frac{2\,\boldsymbol{s}\cdot\boldsymbol{x}^t\, \log(1-\eta\,(\hat{y}^t-y^t)\,X)}{X} + \left(2\,\eta\,(\hat{y}^t-y^t) - \eta^2\,(\hat{y}^t-y^t)^2\,X\right)\hat{y}^t\,.\end{aligned}$$

Denoting by $r \doteq \boldsymbol{s}\cdot\boldsymbol{x}^t$, it suffices to show that $G(\hat{y},y,r) \leq 0$ where (omitting the superscript t)

$$\begin{aligned}G(\hat{y},y,r) = -\frac{2r\log(1-\eta\,(\hat{y}-y)X)}{X} \\ - \left(2\,\eta\,(\hat{y}-y) - \eta^2\,(\hat{y}-y)^2\,X\right)\hat{y} + a\,\|y-\hat{y}\|^2 - b\,\|y-r\|^2\,.\end{aligned}$$

For fixed $\hat{y}$ and y, the function $G(\hat{y},y,r)$ is maximized for

$$r = y - A/(Xb)\,, \quad \text{where} \quad A \doteq \log(1-\eta(\hat{y}-y)X)\,.$$

Plugging in this value, we have $H(\hat{y},y) \doteq G(\hat{y},y,y-A/(Xb))$ where

$$H(\hat{y},y) = -2\,\eta\,(\hat{y}-y)\hat{y} + \eta^2\,(\hat{y}-y)^2\,\hat{y}\,X + a\,(\hat{y}-y)^2 + A^2/(bX^2) - 2\,A\,y/X\,.$$

It is easy to verify that $H(y,y) = \frac{\partial H(y,y)}{\partial y} = 0$. Moreover, notice that

$$\frac{\partial^2 H(y,y)}{\partial y^2} = \frac{(4Xy+2)\,\eta^2 - 4\,b\,\eta + 2\,ab}{b}\,,$$

which is minimized at $y = X$ for $\eta = b/(1+2\,X^2 b)$. Plugging in this value, we have $\frac{\partial^2 H(y,y)}{\partial y^2} \leq 0$ for $a \leq b/(1+2\,X^2 b)$. Now there remains to show that $\hat{y} = y$ is the only maximum of the function $H(\hat{y},y)$. This is easily verified by plugging in

$a = b/(1 + 2\,X^2 b)$ and observing

$$\frac{\partial^2 H(\hat{y}, y)}{\partial y^2} = -\frac{b}{\left(\frac{1}{2} + X^2 b\right)^2 \left(\frac{1}{2} + \left(X + \frac{y}{2} - \frac{\hat{y}}{2}\right) Xb\right)^2} \times$$
$$\left(\frac{1}{2}\left(\frac{1}{2} + X^2 b\right)^2 \log\left(1 + \frac{(y - yh)\,Xb}{1 + 2X^2 b}\right)\right.$$
$$+ Xb\left(\left(X^3 - \left(\frac{y + yh}{2}\right) X^2 + \left(\frac{\hat{y}^2 - y^2}{4}\right) X - \frac{\hat{y}(y - \hat{y})^2}{8}\right) X^2 b^2\right.$$
$$\left.\left. + Xb\left(X^2 - \left(\frac{y + yh}{2}\right) X + \left(\frac{\hat{y}^2 - y^2}{8}\right)\right) + \frac{X}{4} - \frac{y + \hat{y}}{8}\right) \leq 0\,,$$

for $0 \leq \hat{y} \leq X$ and $0 \leq y \leq X$ and

$$\frac{\partial^2 H(y, y)}{\partial y^2} = -\frac{4\,b^2 X(X - y)}{(1 + 2\,X^2 b)^2}\,.$$

Finally, setting $b = c/(X^2)$ for some $c > 0$, we have

$$\sum_{t=1}^{T} \|y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t\|^2 \leq \left(1 + 2\,c\right) \sum_{t=1}^{T} \|y^t - \boldsymbol{s} \cdot \boldsymbol{x}^t\|^2 + \left(2 + \frac{1}{c}\right) XY\,\Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1)\,.$$

Setting $c = 1$ establishes the first bound. Similarly, optimizing for c yields

$$c = \frac{X\sqrt{D}}{\sqrt{2L}}\,.$$

For this choice of c and η as in (4.21), we obtain the second bound.

A.3 Proof Sketch of Claim 1

We can lower-bound the progress of the algorithm as (assuming $\eta \leq 1/(2\,X^2)$),

$$\begin{aligned}
&\Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - \Delta_{\text{RE}}(\boldsymbol{s}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\
&\qquad = 2\,\boldsymbol{s} \cdot \log(\mathbf{1} - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t) \\
&\qquad - \log\left(\boldsymbol{w}^t \cdot (\mathbf{1} - 2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t + \eta\,(\hat{y}^t - y^t)^2\,\boldsymbol{x}^t \odot \boldsymbol{x}^t\right) \\
&\qquad \geq \frac{(r+X)}{X}\,\log\left(\frac{1 - \eta\,(\hat{y}^t - y^t)\,X}{1 + \eta\,(\hat{y}^t - y^t)\,X}\right) + 2\,\log\left(1 + \eta\,(\hat{y}^t - y^t)\,X\right) \\
&\qquad\quad - \log\left(1 - 2\,\eta\,(\hat{y}^t - y^t)\,\hat{y}^t + \eta^2\,(\hat{y}^t - y^t)^2\,X^2\right),
\end{aligned}$$

where $r = \boldsymbol{s} \cdot \boldsymbol{x}^t$. For fixed y and $\hat{y}$, the lower-bound can be maximized for (omitting t)

$$r = y - \frac{1}{Xb}\,\log\left(\frac{1 - \eta\,(\hat{y} - y)\,X}{1 + \eta\,(\hat{y} - y)\,X}\right).$$

Thus, plugging back for r and introducing the new variable $\delta = y - \hat{y}$, it suffices to show that the function $G(a, b, \eta, \delta, \hat{y}) \leq 0$ for the choices of a, b, and η in the claim, and for all $-X \leq \hat{y} \leq X$, where

$$\begin{aligned}
G(a, b, \eta, \delta, \hat{y}) \doteq\ & \log(1 + 2\,\eta\,\delta\,\hat{y} + \eta^2\,\delta^2 X^2) + \frac{1}{4X^2 b}\,\log\left(\frac{1 + \eta\delta X}{1 - \eta\delta X}\right)^2 \\
& - \frac{1}{X}\,(X + \delta + \hat{y})\,\log\left(\frac{1 + \eta\delta X}{1 - \eta\delta X}\right) - 2\,\log(1 - \eta\,\delta X) + a\,\delta^2\,.
\end{aligned}$$

For a fixed δ, this function is maximized at

$$\hat{y}_{\text{opt}} = \frac{X}{\log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right)} - \frac{1}{2}\eta\delta X^2 \quad \text{since} \quad \frac{\partial}{\hat{y}}\,G(a, b, \eta, \delta, \hat{y}_{\text{opt}}) = -\frac{\log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right)^2}{X}\,.$$

Substituting this value for $\hat{y}$, we need to show $H(a,b,\eta) \doteq G(a,b,\eta,\delta,\hat{y}_{\rm opt}) \leq 0$ where

$$H(a,b,\eta) = \log\big(\frac{1+\eta\delta X}{1-\eta\delta X}\big)\Big(\frac{1}{4X^2 b}\,\log\big(\frac{1+\eta\delta X}{1-\eta\delta X}\big) + \frac{\eta\delta X}{2} - \frac{\delta}{X} + \frac{1}{2\eta\delta X}\Big) + \log\left(\frac{\eta\delta X}{\log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right)}\right) + a\,\delta^2 - \log(1-\eta^2\delta^2 X^2) + \log(2) - 1\,.$$

This function can be simplified further by substituting $a = \eta = b/(1+2X^2 b)$. Also, we can introduce two new variables by defining $b = c/X^2$ for some $c > 0$ and $\delta = p\,X$ for $-2 \leq k \leq 2$. AS final step of the proof, there remains to show that the function

$$K(c,p) \doteq \log\left(\frac{1+(2+p)c}{1+(2-p)c}\right)\left(\frac{1}{c}\,\log\left(\frac{1+(2+p)c}{1+(2-p)c}\right) + \frac{pc}{2(2c+1)} - p + \frac{2c+1}{2pc}\right) + \log\left(\frac{pc}{\log\left(\frac{1+(2+p)c}{1+(2-p)c}\right)}\right) + \frac{p^2 c}{2c+1} - \log\big((1+2c)^2 - p^2\big) + \log(2) - 1 \leq 0\,,$$

for all values of $c > 0$ and $-2 \leq p \leq 2$.

Appendix B

B.1 An Iterative Algorithm for Computing the Normalization

Algorithm 7 Iterative algorithm for computing $\lambda_\tau(\hat{\boldsymbol{a}})$ (from [10])

Input: Vector of activations $\hat{\boldsymbol{a}}$, temperature $\tau > 1$
$\mu \leftarrow \max(\hat{\boldsymbol{a}})$
$\tilde{\boldsymbol{a}} \leftarrow \hat{\boldsymbol{a}} - \mu$
while $\tilde{\boldsymbol{a}}$ not converged **do**
  $Z(\tilde{\boldsymbol{a}}) \leftarrow \sum_{i=1}^{k} \exp_\tau(\tilde{a}_i)$
  $\tilde{\boldsymbol{a}} \leftarrow Z(\tilde{\boldsymbol{a}})^{1-\tau}(\hat{\boldsymbol{a}} - \mu\,\mathbf{1})$
Return: $\lambda_\tau(\hat{\boldsymbol{a}}) \leftarrow -\log_\tau \frac{1}{Z(\tilde{\boldsymbol{a}})} + \mu$

B.2 Other Tempered Convex Functions

In the construction of the strict convex function family F_τ we used $F_\tau(x) = \int \log_\tau(x)\,\mathrm{d}\,x$ exploiting the fact that $\log_\tau(x)$ is strictly increasing. We can also define an alternative convex function family $\widetilde{F}_\tau$ by utilizing the convexity (respectively, concavity) of the $\log_\tau$ function for values of $\tau \geq 0$ (respectively, $\tau \leq 0$):

$$\widetilde{F}_\tau(\boldsymbol{y}) = -\frac{1}{\tau}\sum_i(\log_\tau y_i - y_i + 1) = -\frac{1}{\tau\,(1-\tau)}\sum_i(y_i^{1-\tau} - y_i)\,.$$

Note that $\tilde{f}_\tau(\boldsymbol{y}) \doteq \nabla \widetilde{F}_\tau(\boldsymbol{y}) = \frac{1-\boldsymbol{y}^{-\tau}}{\tau}$ and $\nabla^2 \widetilde{F}_\tau(\boldsymbol{y}) = \operatorname{diag}(\boldsymbol{y}^{-(1+\tau)})$, thus $\widetilde{F}_\tau$ is indeed a strictly convex function. The following proposition shows that the Bregman divergence induced by the original and the alternate convex function are related by a temperature shift:

Proposition 9. *For the Bregman divergence induced by the convex function $\widetilde{F}_\tau$, we have*

$$\forall \boldsymbol{y},\, \hat{\boldsymbol{y}} \in \mathbb{R}_+^k : \quad \Delta_{\widetilde{F}_\tau}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \;=\; \frac{1}{\tau} \sum_i (\log_\tau \hat{y}_i - \log_\tau y_i + (y_i - \hat{y}_i)\, \hat{y}_i^{-\tau}) \;=\; \Delta_{F_{\tau+1}}(\boldsymbol{y}, \hat{\boldsymbol{y}})\,.$$

The $\widetilde{F}_\tau$ function is also related to the negative Tsallis entropy over the probability vector $\boldsymbol{y} \in S^k$ defined as

$$-H_\tau^{\text{Tsallis}}(\boldsymbol{y}) = \frac{1}{1-\tau}\Big(1 - \sum_i y_i^\tau\Big) = -\sum_i y_i \log_\tau \frac{1}{y_i}\,.$$

Note that $(-H_\tau^{\text{Tsallis}} - (1-\tau)\,\widetilde{F}_{1-\tau})$ is an affine function. Thus, the Bregman Divergence induced by $\widetilde{F}_\tau$, and the one induced by $-H_\tau^{\text{Tsallis}}$ are also equivalent up to a scaling and a temperature shift. Thus, both functions F_τ and $\widetilde{F}_\tau$ can be viewed as some generalized negative entropy functions. Note that the Bregman divergence induced by $-H_\tau^{\text{Tsallis}}$ is fundamentally different from the Tsallis divergence over the simplex, defined as

$$\Delta_\tau^{\text{Tsallis}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i y_i \log_\tau \frac{\hat{y}_i}{y_i} = \sum_i y_i^\tau\, (\log_\tau y_i - \hat{y}_i)\,.$$

B.3 Convexity of the Tempered Matching Loss

The convexity of the loss function $\Delta_{F_\tau}\Big(\boldsymbol{y}, \exp_\tau(\hat{\boldsymbol{a}} - \lambda_\tau(\hat{\boldsymbol{a}})\,\mathbf{1}\Big)$ with $\tau \geq 1$ for $\hat{\boldsymbol{a}} \in \mathbb{R}^k$ immediately follows from the definition of the matching loss. A more

subtle case occurs when $0 \leq \tau < 1$. Note that the range of the combined function $\log_\tau \circ \exp_\tau$ does not cover all $\mathbb{R}^k$ as the $\log_\tau$ function is bounded from below by $-\frac{1}{1-\tau}$. Therefore, $\text{range}(\log_\tau \circ \exp_\tau) = \{\boldsymbol{a}' \in \mathbb{R}^k \mid -\frac{1}{1-\tau} \leq \boldsymbol{a}'\}$.

Remark 2. *The normalization function $\lambda_\tau(\hat{\boldsymbol{a}})$ satisfies:*

$$\lambda_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) = \lambda_\tau(\hat{\boldsymbol{a}}) + b \quad \textit{for any } \tau \geq 0 \textit{ and } b \in \mathbb{R}\,.$$

Proof. Note that

$$\sum_i \exp_\tau\Big((\hat{a}_i{+}b){-}\lambda_\tau(\hat{\boldsymbol{a}}{+}b\,\mathbf{1})\Big) = \sum_i \exp_\tau\Big(\hat{a}_i{-}\underbrace{(\lambda_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) - b)}_{=\lambda_\tau(\hat{\boldsymbol{a}})}\Big) = 1 \quad \text{for} \quad \forall \hat{\boldsymbol{a}} \in \mathbb{R}^k\,.$$

The claim follows immediately. □

Proposition 10. *The loss function $\Delta_{F_\tau}\Big(\boldsymbol{y}, \exp_\tau(\hat{\boldsymbol{a}} - \lambda_\tau(\hat{\boldsymbol{a}})\,\mathbf{1})$ for $0 \leq \tau < 1$ is convex for*

$$\hat{\boldsymbol{a}} \in \{\boldsymbol{a}' + \mathbb{R}\,\mathbf{1} \mid -\frac{1}{1-\tau} \leq \boldsymbol{a}' \;\; \textit{and} \;\; \boldsymbol{a}' \cdot \mathbf{1} = 0\}\,.$$

Proof. Using the definition of the dual function $\check{F}^*$ and its derivative $\check{f}^*$, we can write

$$\begin{aligned}
\Delta_{F_\tau}(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= F_\tau(\boldsymbol{y}) - F_\tau(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}}) \cdot f_\tau(\hat{\boldsymbol{y}}) \qquad \Big(\hat{\boldsymbol{y}} = \exp_\tau(\hat{\boldsymbol{a}} - \lambda_\tau(\hat{\boldsymbol{a}})\,\mathbf{1})\Big) \\
&= F_\tau(\boldsymbol{y}) - F_\tau(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}}) \cdot \log_\tau \circ \exp_\tau(\hat{\boldsymbol{a}} - \lambda(\hat{\boldsymbol{a}})\,\mathbf{1}) \\
&= F_\tau(\boldsymbol{y}) - F_\tau(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}}) \cdot \hat{\boldsymbol{a}} \qquad \Big((\boldsymbol{y} - \hat{\boldsymbol{y}}) \cdot \mathbf{1} = 1 - 1 = 0\Big) \\
&= \underbrace{F_\tau(\boldsymbol{y}) - \boldsymbol{y} \cdot (\check{f}_\tau^*)^{-1}(\boldsymbol{y})}_{-\check{F}_\tau^*((\check{f}_\tau^*)^{-1}(\boldsymbol{y}))} + \boldsymbol{y} \cdot (\check{f}_\tau^*)^{-1}(\boldsymbol{y}) \underbrace{- F_\tau(\hat{\boldsymbol{y}}) + \hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{a}}}_{\check{F}_\tau^*(\hat{\boldsymbol{a}})} - \boldsymbol{y} \cdot \hat{\boldsymbol{a}} \\
&= \check{F}_\tau^*(\hat{\boldsymbol{a}}) - \check{F}_\tau^*((\check{f}_\tau^*)^{-1}(\boldsymbol{y})) - (\hat{\boldsymbol{a}} - (\check{f}_\tau^*)^{-1}(\boldsymbol{y})) \cdot \boldsymbol{y} \\
&= \Delta_{\check{F}_\tau^*}(\hat{\boldsymbol{a}}, (\check{f}_\tau^*)^{-1}(\boldsymbol{y}))\,.
\end{aligned}$$

Note that the transition from the second line to the third line requires that the assumption $-\frac{1}{1-\tau} \leq \hat{\boldsymbol{a}}$ holds. The dual function $\check{F}^*_\tau$ satisfies

$$\check{F}^*_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) = \lambda_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) + \frac{1}{2-\tau} \sum_i \exp_\tau\Big((\hat{a}_i + b) - \lambda_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1})\Big)^{2-\tau} = \check{F}^*_\tau(\hat{\boldsymbol{a}}) + b\,.$$

Taking the derivative w.r.t. $\hat{\boldsymbol{a}}$ yields

$$\check{f}^*_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) = \check{f}^*_\tau(\hat{\boldsymbol{a}})\,.$$

Additionally,

$$\begin{aligned}
&\Delta_{\check{F}^*_\tau}(\hat{\boldsymbol{a}} + b\,\mathbf{1}, (\check{f}^*_\tau)^{-1}(\boldsymbol{y})) \\
&= \check{F}^*_\tau(\hat{\boldsymbol{a}} + b\,\mathbf{1}) - \check{F}^*_\tau((\check{f}^*_\tau)^{-1}(\boldsymbol{y})) - (\hat{\boldsymbol{a}} + b\,\mathbf{1} - (\check{f}^*_\tau)^{-1}(\boldsymbol{y})) \cdot \boldsymbol{y} = \Delta_{\check{F}^*_\tau}(\hat{\boldsymbol{a}}, (\check{f}^*_\tau)^{-1}(\boldsymbol{y}))\,.
\end{aligned}$$

The claim follows by considering the range of $\log_\tau \circ \exp_\tau$ and the invariance of the Bregman divergence induced by $\check{F}^*_\tau$ along $\mathbb{R}\,\mathbf{1}$. □

B.4 Derivatives of Lagrangian and the Bi-tempered Matching Loss

The gradient of $\lambda_\tau(\hat{\boldsymbol{a}})$ w.r.t. $\hat{\boldsymbol{a}}$ can be calculated by taking the partial derivative of both sides of the equality $1 = \sum_j \exp_\tau(\hat{a}_j - \lambda_\tau(\hat{\boldsymbol{a}}))$ w.r.t. $\hat{a}_i$:

$$\begin{aligned}
0 &= \sum_j \exp_\tau(\hat{a}_j - \lambda_\tau(\hat{\boldsymbol{a}}))^\tau \left(\delta_{ij} - \frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i}\right) \\
&= \exp_\tau\Big(\hat{a}_i - \lambda_\tau(\hat{\boldsymbol{a}})\Big)^\tau - \frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i} \sum_j \exp_\tau(\hat{a}_j - \lambda_\tau(\hat{\boldsymbol{a}}))^\tau, \qquad \text{(B.1)}
\end{aligned}$$

where $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$. Therefore $\frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i} = \frac{\exp_\tau(\hat{a}_i - \lambda_\tau(\hat{\boldsymbol{a}}))^\tau}{Z_\tau}$, where $Z_\tau = \sum_j \exp_\tau(\hat{a}_j - \lambda_\tau(\hat{\boldsymbol{a}}))^\tau$. This concludes that $\frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i}$ is the "τ-escort distribution" of the distribution $\hat{y}_i = \exp(\hat{a}_i - \lambda_\tau(\hat{\boldsymbol{a}}))$.

Similarly, the second derivative of $\lambda_\tau(\hat{\boldsymbol{a}})$ can be calculated by repeating the derivation on (B.1):

$$\frac{\partial^2 \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i \partial \hat{a}_j} = \frac{1}{Z_\tau} \sum_{j'} \tau \, \exp_\tau\left(\hat{a}_{j'} - \lambda_\tau(\hat{\boldsymbol{a}})\right)^{2t-1} \left(\delta_{ij'} - \frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_i}\right)\left(\delta_{jj'} - \frac{\partial \lambda_\tau(\hat{\boldsymbol{a}})}{\partial \hat{a}_j}\right).$$

Although not immediately obvious from the second derivative, it is easy to show that $\lambda_\tau(\hat{\boldsymbol{a}})$ is in fact convex in $\hat{\boldsymbol{a}}$. Also the derivative of the loss $L_{\tau_1}^{\tau_2}(\hat{\boldsymbol{a}} | \, \boldsymbol{y})$ w.r.t. $\hat{a}_i$ (expressed in terms of $\boldsymbol{y}$ and $\hat{\boldsymbol{y}} = \exp_{\tau_2}(\hat{\boldsymbol{a}} - \lambda_{\tau_2}(\hat{\boldsymbol{a}})\,\mathbf{1})$) becomes

$$\begin{aligned}
\frac{\partial L_{\tau_1}^{\tau_2}}{\partial \hat{a}_i} &= \sum_j \frac{\partial}{\partial \hat{y}_j}\Big(y_j \log_{\tau_1} y_j - y_j \log_{\tau_1} \hat{y}_j - \frac{1}{2-\tau_1} y_j^{2-\tau_1} + \frac{1}{2-\tau_1} \hat{y}_j^{2-\tau_1}\Big) \frac{\partial \hat{y}_j}{\partial \hat{a}_i} \\
&= \sum_j (\hat{y}_j - y_j)\, \hat{y}_j^{\tau_2 - \tau_1} \left(\delta_{ij} - \frac{\hat{y}_i^{\tau_2}}{\sum_{j'} \hat{y}_{j'}^{\tau_2}}\right).
\end{aligned}$$

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Ethan Akin. *The geometry of population genetics*, volume 31 of *Lecture Notes in Biomathematics*. Springer-Verlag, Berlin-New York, 1979.

[3] S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*. American Mathematical Society: Providence, RI, USA, 2000.

[4] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[5] E. Amid, M. K. Warmuth, R. Anil, and K. Tomer. Robust bi-tempered logistic loss based on Bregman divergences. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'19, 2019.

[6] Ehsan Amid and Manfred K. Warmuth. Divergence-based motivation for online EM and combining hidden variable models. *Uncertainty in Artificial Intelligence (UAI)*, 2020.

[7] Ehsan Amid and Manfred K. Warmuth. An implicit form of krasulina's k-PCA update without the orthonormality constraint. *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

[8] Ehsan Amid and Manfred K. Warmuth. Interpolating between gradient descent and exponentiated gradient using reparameterized gradient descent. *arXiv preprint arXiv:2002.10487*, 2020.

[9] Ehsan Amid and Manfred K. Warmuth. Winnowing with gradient descent. *Conference on Learning Theory (COLT)*, 2020.

[10] Ehsan Amid, Manfred K. Warmuth, and Sriram Srinivasan. Two-temperature logistic regression based on the Tsallis divergence. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS 19)*, 2019.

[11] Katy S Azoury and Manfred K Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

[12] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

[13] Michéle Basseville. Divergence measures for statistical data processing. *Signal Processing*, 93(4):621–633, 2013.

[14] Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[15] Jean-Daniel Boissonnat, Frank Nielsen, and Richard Nock. Bregman Voronoi diagrams. *Discrete & Computational Geometry*, 44(2):281–307, 2010.

[16] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[17] Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, University of Pennsylvania, November 2005.

[18] Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007.

[19] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[20] Andrzej Cichocki and Shun-ichi Amari. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010.

[21] Andrzej Cichocki, Rafal Zdunek, Anh-Huy Phan, and Shun-ichi Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Wiley, first edition, 2009.

[22] I. Csiszár. Information-type measures of difference of probability distributions and indirect observations. In *Studia Scientiarum Mathematicarum Hungarica*, volume 2, pages 229–318, 1967.

[23] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[25] M. Derezinski and M. K. Warmuth. The limits of squared Euclidean distance regularization. In *Advances in Neural Information Processing Systems 27*, NIPS'14, pages 2807–2815, Cambridge, MA, USA, 2014.

[26] Inderjit S Dhillon and Joel A Tropp. Matrix nearness problems with Bregman divergences. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1120–1146, 2008.

[27] Nan Ding. *Statistical machine learning in the t-exponential family of distributions*. PhD thesis, Purdue University, 2013.

[28] Nan Ding and S. V. N. Vishwanathan. t-logistic regression. In *Proceedings of the 23th International Conference on Neural Information Processing Systems*, NIPS'10, pages 514–522, Cambridge, MA, USA, 2010.

[29] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[30] U. Ghai, E. Hazan, and S. Singer. Exponentiated gradient meets gradient descent. *arXiv preprint arXiv:1902.01903*, 2019.

[31] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems 30*, NIPS'17, pages 6151–6159, 2017.

[32] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *International Conference on Learning Representations (ICLR)*, 2017.

[33] David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

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

[35] D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, November 1999.

[36] David P Helmbold, Sandra Panizza, and Manfred K Warmuth. Direct and indirect algorithms for on-line learning of disjunctions. *Theoretical Computer Science*, 284(1):109–142, 2002.

[37] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer-Verlag Berlin Heidelberg, first edition, 2001.

[38] J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

[39] J. Kivinen, M. K. Warmuth, and P. Auer. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97:325–343, December 1997.

[40] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.

[41] Jyrki Kivinen, Manfred K Warmuth, and Babak Hassibi. The p-norm generalization of the LMS algorithm for adaptive filtering. *IEEE Transactions on Signal Processing*, 54(5):1782–1793, 2006.

[42] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[43] Brian Kulis and Peter L. Bartlett. Implicit online learning. In *ICML*, 2010.

[44] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[45] Z. Liao and R. Couillet. The dynamics of learning: A random matrix approach. *arXiv preprint 1805.11917*, 2018.

[46] N Littlestone and MK Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[47] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[48] Philip M Long and Rocco A Servedio. Random classification noise defeats all convex potential boosters. In *Proceedings of the 25th international conference on Machine learning*, pages 608–615. ACM, 2008.

[49] Jan Naudts. deformed exponentials and logarithms in generalized thermostatistics. *physica a*, 316:323–334, 2002.

[50] A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley & Sons, New York, 1983.

[51] Jiazhong Nie, Wojciech Kotłowski, and Manfred K Warmuth. Online PCA with optimal regret. *The Journal of Machine Learning Research*, 17(1):6022–6070, 2016.

[52] Frank Nielsen and Richard Nock. On the centroids of symmetrized Bregman divergences. *arXiv preprint arXiv:0711.3242*, 2007.

[53] Maxim Raginsky and Jake Bouvrie. Continuous-time stochastic mirror descent on a network: Variance reduction, consensus, convergence. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6793–6800. IEEE, 2012.

[54] Garvesh Raskutti and Sayan Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

[55] M. D. Reid and R. C. Williamson. Surrogate regret bounds for proper losses. In *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, pages 897–904, 2009.

[56] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[57] William H Sandholm. *Population games and evolutionary dynamics*. MIT Press, 2010.

[58] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[59] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[60] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

[61] Ambuj Tewari and Peter L Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8(May):1007–1025, 2007.

[62] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487, 1988.

[63] Tim Van Erven and Peter Harremos. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

[64] Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. In *Advances in Neural Information Processing Systems 32*, pages 2968–2979, 2019.

[65] Baba C Vemuri, Meizhu Liu, Shun-Ichi Amari, and Frank Nielsen. Total Bregman divergence and its applications to DTI analysis. *IEEE Transactions on medical imaging*, 30(2):475–483, 2010.

[66] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[67] S.V.N. Vishwanathan and M.K. Warmuth. Leaving the span. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT 05)*, Bertinoro, Italy, June 2005. Springer.

[68] Volodimir G Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, pages 371–386. Morgan Kaufmann Publishers Inc., 1990.

[69] M. K. Warmuth and A. Jagota. Continuous and discrete time nonlinear gradient descent: relative loss bounds and convergence. In R. Greiner E. Boros, editor, *Electronic Proceedings of Fifth International Symposium on Artificial Intelligence and Mathematics*. Electronic,http://rutcor.rutgers.edu/˜amai, 1998.

[70] M. K. Warmuth, W. Kotłowski, and S. Zhou. Kernelization of matrix updates. *Journal of Theoretical Computer Science*, 558:159–178, 2014. Special issue for the 23nd International Conference on Algorithmic Learning Theory (ALT'12).

[71] Manfred K. Warmuth. Winnowing subspaces. In *Proceedings of the 24th International Conference on Machine Learning*, ICML'07, pages 999–1006, New York, NY, USA, 2007. ACM.

[72] Robert C. Williamson, Elodie Vernet, and Mark D. Reid. Composite multiclass losses. *Journal of Machine Learning Research*, 17(223):1–52, 2016.

[73] Blake Woodworth, Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

[74] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[75] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, pages 8778–8788, 2018.

[76] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.