Abstract

The exponential family of distributions play an important role in statistics and machine learning. They underlie numerous models such as logistic regression and probabilistic graphical models. However, exponential family based probabilistic models are vulnerable to outliers. This paper focuses on the classification problem and propose t-logistic regression. It replaces the exponential family in logistic regression by a more generalized distribution family, namely t-exponential family. We demonstrate the robustness of t-logistic regression both theoretically and empirically. We also show that the algorithm is empirically stable although the empirical risk may have multiple local minima.

1. INTRODUCTION

Consider the classic machine learning problem of binary classification: we are given \( m \) training data points \( \{x_1, \ldots, x_m\} \) and their corresponding labels \( \{y_1, \ldots, y_m\} \), with \( x_i \) drawn from some vector space \( \mathcal{X} \) and \( y_i \in \{+1, -1\} \). The task is to learn a function \( f : \mathcal{X} \to \{+1, -1\} \) which can predict the labels on unseen data. In this paper, we focus on linear models: \( f(x) := \text{sign} \left( \langle \Phi(x), \theta \rangle \right) \). Here \( \Phi \) is a feature map, \( \theta \) are the parameters of the model, \( \langle \cdot, \cdot \rangle \) denotes an inner product, and \( \text{sign}(z) = +1 \) if \( z > 0 \) and \(-1\) otherwise.
One way to learn $\theta$ is to define a loss function $l(x, y, \theta)$ and minimize the averaged loss, or empirical risk:

$$\min_{\theta} R_{\text{emp}}(\theta) := \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, \theta).$$

In order to prevent overfitting to the training data, it is customary to add a regularizer $\Omega(\theta)$ to $R_{\text{emp}}(\theta)$ and minimize the regularized risk:

$$\min_{\theta} J(\theta) := \lambda \Omega(\theta) + R_{\text{emp}}(\theta).$$

Here $\lambda$ is a scalar which trades off the importance of the regularizer and the empirical risk. While a variety of regularizers are commonly used (see e.g. (Teo et al., 2010)), we will restrict our attention to the $L_2$-regularizer:

$$\Omega(\theta) = \frac{1}{2} \|\theta\|_2^2.$$

Let $\hat{a} = \langle \Phi(x), \theta \rangle$ and $u(x, y, \theta) := y \cdot \hat{a}$ denote the margin of $(x, y)$. Where it is clear from context, we will use $u$ to denote $u(x, y, \theta)$ and $u_i$ to denote $u(x_i, y_i, \theta)$. Note that $u > 0$ if, and only if, $f(x) = y$, that is, $x$ is correctly classified. Therefore, a natural loss function to define is the 0-1 loss:

$$l(x, y, \theta) = \begin{cases} 0 & \text{if } u > 0 \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

Unfortunately, the 0-1 loss is non-convex, non-smooth, and it is NP-hard to even approximately minimize the empirical risk with this loss (Ben-David et al., 2003). Therefore, a lot of research effort has been directed towards finding surrogate losses which are computationally tractable. In particular, convex loss functions are in vogue mainly because the regularized risk minimization problem can be solved efficiently with readily available tools (Boyd and Vandenberghe, 2004). Table 1 summarizes a few popular convex losses and Figure 1 contrasts them with the 0-1 loss.

Table 1: Some popular convex losses used for binary classification. The loss functions are plotted in Figure 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Loss Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hinge</td>
<td>$l(x, y, \theta) = \max(0, 1 - u)$</td>
</tr>
<tr>
<td>Exponential</td>
<td>$l(x, y, \theta) = \exp(-u)$</td>
</tr>
<tr>
<td>Logistic</td>
<td>$l(x, y, \theta) = \log(1 + \exp(-u))$</td>
</tr>
</tbody>
</table>

Despite many successes of the binary classification algorithms based on convex losses, as (Long and Servedio, 2010; Manwani and Sastry, 2012) point out, those algorithms are not noise-tolerant\footnote{1. Although, the analysis of Long and Servedio (2010) is carried out in the context of boosting, we believe, the results hold for a larger class of algorithms which minimize a regularized risk with a convex loss function.}
Figure 1: Some commonly used convex surrogate loss functions, including hinge loss, logistic loss, and exponential loss, for binary classification.

(see Section 2.1). Intuitively, as can be seen from Figure 1, the convex loss functions grow at least linearly as $u \in (-\infty, 0)$, which causes data with $u \ll 0$ to become too important.

There has been some recent and not-so-recent work on using non-convex loss functions to alleviate the above problem. Although these non-convex losses are empirically more robust, they also lose some key advantages of convex losses. For instance, finding the global optimum becomes very hard because the empirical risk may be multimodal. Unlike certain convex losses, such as logistic regression (see Section 2.2), those non-convex losses do not have a proper probabilistic interpretation. The probabilistic interpretation is important to generalize these losses to more complex settings, such as modeling interacting factors where probabilistic graphical models are widely applied (Wainwright and Jordan, 2003, 2008).

In this paper, we propose to investigate a non-convex loss function which is firmly grounded in probability theory. By extending logistic regression from the exponential family to the $t$-exponential family\(^2\), a natural extension of exponential family of distributions studied in statistical physics (Naudts, 2002, 2004a,b,c; Tsallis, 1988) (reviewed in Section 2.3), we obtain the $t$-logistic regression. Our main idea is to use $t$-exponential family to replace the exponential family for modeling the conditional likelihood of the examples. We demonstrate the robustness of $t$-logistic regression both theoretically and empirically. We also show that the algorithm is empirically stable under random initialization. Furthermore, as shown in (Ding, 2013), our idea can be generalized to more complicated probabilistic models, e.g. $t$-conditional random fields.

The structure of the rest of the paper is as follows: Section 2 reviews some related background material, including noise tolerance of convex losses, probabilistic interpretation of logistic regression, and the $t$-exponential family of distributions. Section 3 introduces the $t$-logistic regression for binary classification. Section 4 investigates three properties of binary $t$-logistic regression:

\(^2\) Also known as the $q$-exponential family or the Tsallis distribution in statistical physics. C. Tsallis is one of pioneers of nonextensive entropy and generalized exponential family.
Bayes-risk consistency, robustness, and multiple local minima. Section 5 introduces the multi-class $t$-logistic regression. Section 6 discusses the optimization algorithms for $t$-logistic regression. Section 7 conducts extensive experiments on a variety of publicly available binary and multiclass datasets and shows that $t$-logistic regression is robust in the presence of label noise and does not stick in local minima in practice.

2. BACKGROUND

In this section, we present some existing literature and background material that is used later in this paper. We will first review the famous example proposed in (Long and Servedio, 2010) which shows that uniform random label noise defeats all convex classifiers. Next, we review logistic regression, discuss its probabilistic interpretation and relation to exponential family of distributions. Finally, we review $t$-exponential family of distributions.

2.1 Noise Tolerance of Convex Losses

Convexity is a very attractive property because it ensures that the regularized risk minimization problem has a unique global optimum\(^3\) (Boyd and Vandenberghe, 2004). However, as was recently shown by (Long and Servedio, 2010), learning algorithms based on convex loss functions are not robust to noise. In (Long and Servedio, 2010), the authors constructed an interesting dataset to show that convex losses are not tolerant to uniform label noise (label noise is added by flipping a portion of the labels of the training data). In their dataset, each data point has a 21-dimensional feature vector and plays one of three possible roles: large margin examples (25%, $x_{1,2,\ldots,21} = y$); pullers (25%, $x_{1,\ldots,11} = y$, $x_{12,\ldots,21} = -y$); and penalizers (50%, Randomly select and set 5 of the first 11 coordinates and 6 out of the last 10 coordinates to $y$, and set the remaining coordinates to $-y$). Note that all the data has the same magnitude in terms of $L_1$, $L_2$, $L_\infty$ norm.

This dataset is illustrated in Figure 2. We use the red blobs to represent the points with label +1, and the blue blobs to represent the points with label −1. Each blob plays one of the three roles as marked on the figure. Without adding label noise, the black double arrow (N-S) is the optimal classifier of the convex losses, which classifies the clean data perfectly. However, if we add 10% label noise into the dataset (represented by narrow red or blue circles surrounding the blue or red blobs), the optimal classifier of the convex losses changes to the red double arrow (NW-SE). Obviously, the new classifier is no longer able to distinguish the penalizers.

We can intuitively see the reason from the shape of the convex loss functions. According to Figure 1, the convex losses grow at least linearly with slope $|l'(0)|$ as $u \in (-\infty, 0)$, which introduces an extremely large loss from the data point with $u \ll 0$. Therefore, the flipped large margin examples in Figure 2 dramatically increase the empirical risk of the black classifier, which becomes larger than that of the red classifier.

Since convex losses are non-robust against random label noise, many nonconvex losses have been investigated to improve the robustness of the classifier. We list some but not all commonly used nonconvex losses in Table 2. However, those non-convex losses have their own problems. First of all, although the non-convex losses are empirically more robust, they also lose some key advantages of convex losses, especially they may stuck into local minima of their empirical risk. More importantly, over the past decades, probabilistic graphical models (Wainwright and Jordan,

---

\(^3\) By unique global optimum, we mean the uniqueness of minimum objective.
large margin (25%), puller (25%), penalizer (50%). The black double arrow represents the true classifier. The red double arrow represents the optimal classifier of convex losses when 10% of data labels are flipped (represented by the circles surrounding the blobs). The red double arrow is no longer able to classify the penalizers.

2003, 2008) have been widely used as powerful and efficient tools to model interacting factors in multivariate data. However, none of those losses have a proper probabilistic interpretation (e.g. see Section 2.2), which largely limits their generalization to more complicated applications.

Table 2: A few examples of non-convex losses for binary classification. The loss functions are plotted in Figure 3. erf is error function.

<table>
<thead>
<tr>
<th>Name</th>
<th>Loss Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probit</td>
<td>( l(x, y, \theta) = 1 - \text{erf}(u) )</td>
</tr>
<tr>
<td>Ramp</td>
<td>( l(x, y, \theta) = \min(2, \max(0, 1 - u)) )</td>
</tr>
<tr>
<td>Savage</td>
<td>( l(x, y, \theta) = \frac{4}{(1 + \exp(u))^2} )</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>( l(x, y, \theta) = \frac{2}{1 + \exp(u)} )</td>
</tr>
</tbody>
</table>

2.2 Logistic Regression and Exponential Family of Distributions

In contrast, logistic loss, the loss function of logistic regression, is well motivated from a probabilistic perspective. As shown in (Lafferty et al., 2001; Sutton and McCallum, 2006) (also see Section 5 in (Ding, 2013)), its generalization to probabilistic graphical models, e.g. conditional random
fields, will be natural and convenient. In this subsection, we briefly review logistic regression and its relation to exponential family of distributions (Barndorff-Nielsen, 1978).

In statistics, the data points in a dataset are typically assumed to be independently identically distributed (i.i.d.), which allows us to write the conditional likelihood of the entire datasets \((X, y) = \{(x_i, y_i)\}\) with \(i = 1, \ldots, m\) as,

\[
p(y \mid X, \theta) = \prod_{i=1}^{m} p(y_i \mid x_i, \theta).
\]  

(2)

To avoid overfitting to the data, we add a prior \(p(\theta)\) on the parameter \(\theta\). Therefore, according to the Bayes rule, the posterior of \(\theta\) is obtained,

\[
p(\theta \mid y, X) = p(y \mid X, \theta)p(\theta)/p(y \mid X),
\]

and the maximum a-posteriori (MAP) estimate of \(\theta\) is obtained by minimizing,

\[
- \log p(\theta \mid y, X) = - \sum_{i=1}^{m} \log p(y_i \mid x_i; \theta) - \log p(\theta) + \text{const.}
\]  

(3)

where \(\log p(y \mid X)\) is neglected since it is independent of \(\theta\). \(- \log p(\theta)\) serves as the regularizer.

In logistic regression, \(p(y \mid x, \theta)\) is modeled using the exponential family of distributions. The exponential family of distributions (Barndorff-Nielsen, 1978) of a set of random variables \(z\) is a

---

**Figure 3**: Some commonly used non-convex loss functions, including ramp loss, sigmoid loss, and Savage loss, for binary classification. We omit the probit loss because it is very close to sigmoid loss.
parametric distribution family defined as:

\[ p(z; \theta) := \exp \left( \langle \Phi(z), \theta \rangle - G(\theta) \right), \]

where \( \langle \cdot, \cdot \rangle \) is the inner product, \( \Phi(z) \) is a map from \( z \in Z \) to the sufficient statistics, \( \theta \) is commonly referred to as the natural parameter, and it lives in the space dual to \( \Phi(z) \) (see Theorem ??). \( G(\theta) \) is a normalizer, also known as the log-partition function, which ensures that \( p(z; \theta) \) is properly normalized,

\[ G(\theta) = \log \int_Z \exp \left( \langle \Phi(z), \theta \rangle \right) \, dz. \]

Exponential family of distributions has many important properties and applications. Since many of them are non-trivial, we will review and compare them with their generalizations later in Section 2.3.

For binary logistic regression,

\[ p(y|x; \theta) = \exp \left( \langle \Phi(x, y), \theta \rangle - G(x; \theta) \right), \]

where \( \Phi(x, y) = \frac{1}{2} y \Phi(x) \), so that

\[ G(x; \theta) = \log \left( \exp \left( \frac{1}{2} \langle \Phi(x), \theta \rangle \right) + \exp \left( -\frac{1}{2} \langle \Phi(x), \theta \rangle \right) \right). \]

The function \( l(x, y; \theta) := -\log p(y|x; \theta) \) is the logistic loss of the data \( (x, y) \), because

\[ l(x, y; \theta) = -\frac{y}{2} \langle \Phi(x), \theta \rangle + \log \left( \exp \left( \frac{1}{2} \langle \Phi(x), \theta \rangle \right) + \exp \left( -\frac{1}{2} \langle \Phi(x), \theta \rangle \right) \right) = \log \left( 1 + \exp(-y \langle \Phi(x), \theta \rangle) \right). \]

2.3 \( \phi \)-Exponential Family of Distributions

The convexity of logistic regression is essentially because it uses exponential family to model the conditional distribution. The thin-tailed nature of the exponential family makes it unsuitable for designing robust algorithms against noisy data. In the past several decades, effort has also been devoted to develop alternate, generalized distribution families in statistics (Grunwald and Dawid, 2004; Shalizi, 2007), statistical physics (Tsallis, 1988; Naudts, 2002), and most recently in machine learning (Sears, 2008). Of particular interest to us is the \( t \)-exponential family, which was first proposed by Tsallis and co-workers (Tsallis, 1988; Sousa and Tsallis, 1994; Tsallis et al., 1998). It is a special case of the more general \( \phi \)-exponential family of Naudts (Naudts, 2002, 2004a,b,c).

In this subsection, we begin by reviewing the generalized \( \log_\phi \) and \( \exp_\phi \) functions which were introduced in statistical physics by (Naudts, 2002, 2004a,b,c). Then, these generalized exponential functions are used to define \( \phi \)-exponential family of distributions (Naudts, 2004c) and \( t \)-exponential family of distributions as special cases.

---

4. Traditionally exponential family distributions are written as \( p(z; \theta) := p_0(z) \exp \left( \langle \Phi(z), \theta \rangle - G(\theta) \right) \). For ease of exposition we ignore the base measure \( p_0(z) \) in this paper.
2.3.1 Φ-LOGARITHM

The φ-logarithm, \( \log_\phi \), is defined as follows:

**Definition 2.1 (φ-logarithm (Naudts, 2002, 2004a,b,c))** Let \( \phi : [0, +\infty) \to [0, +\infty) \) be strictly positive and non-decreasing on \((0, +\infty)\). Define \( \log_\phi \) via

\[
\log_\phi(x) := \int_1^x \frac{1}{\phi(y)} \, dy \tag{7}
\]

If this integral converges for all finite \( x > 0 \), then \( \log_\phi \) is called a φ-logarithm.

To see that this definition generalizes \( \log \), simply set \( \phi(y) = y \). Clearly, the gradient of \( \log_\phi(x) \) is \( 1/\phi(x) \) from which it follows that \( \log_\phi \) is a concave increasing function. Furthermore, \( \log_\phi \) is negative on \((0, 1)\), positive on \((1, +\infty)\), with \( \log_\phi(1) = 0 \). Of course the integral may diverge at \( x = 0 \). All these are properties of the familiar \( \log \) function. An important example is

**Example 1 (t-logarithm)** Let \( \phi(x) = x^t, t > 0 \). Then

\[
\log_t(x) := \begin{cases} 
\log(x) & \text{if } t = 1 \\
\frac{x^{1-t} - 1}{1-t} & \text{otherwise} 
\end{cases} \tag{8}
\]

and

\[
\frac{d}{dx} \log_t(x) = x^{-t}. \tag{9}
\]

Figure 4 visualizes \( \log_t \) for various values of \( t \) and contrasts it with the familiar \( \log \).

Figure 4: The left figure depicts \( \log_t \) for the various values of \( t \) indicated. The right figure zooms in to better depict the interval \([0, 1]\) in which \( \log_t \) are negative.

---

5. Note that throughout this paper, \( \log_\phi \) or \( \log_t \) are defined in (7) and (8). The subscripts do not represent the log base.
2.3.2 Φ-Exponential

The inverse of \( \log_\phi \) is the \( \phi \)-exponential function, denoted \( \exp_\phi \). When \( \log_\phi \) takes on a finite value, this is well defined. But, unlike \( \log \), there is no guarantee that \( \log_\phi \) takes on all values in \( \mathbb{R} \). Therefore, define \( \exp_\phi(x) = 0 \) if \( x \) is less than every element of \( \text{range}(\log_\phi) \) and \( \exp_\phi(x) = +\infty \) if \( x \) is larger than \( \text{range}(\log_\phi) \). Properties of \( \exp_\phi \), such as convexity, mirror those of \( \log_\phi \) (Naudts, 2004a, 2002). A key difference involves the fact that \( \exp \) is the only non-trivial function which is its own derivative. However \( \exp_\phi \) has the following property:

\[
\frac{d}{dx} \exp_\phi(x) = \phi(\exp_\phi(x)).
\] (10)

**Example 2 (\( t \)-exponential)** Let \( [x]_+ \) be \( x \) if \( x > 0 \) and 0 otherwise. Continuing with \( \phi(x) = x^t \), \( t > 0 \), we have

\[
\exp_t(x) = \begin{cases} 
\exp(x) & \text{if } t = 1 \\
\left[1 + (1 - t)x\right]_{+}^{\frac{1}{1-t}} & \text{otherwise.}
\end{cases}
\] (11)

Elementary calculus shows that

\[
\frac{d}{dx} \exp_t(x) = \exp_t(x)^t
\] (12)

![Figure 5](image_url)

Figure 5: The left figure depicts \( \exp_t \) for the various values of \( t \) indicated. The right figure zooms in to better depict when \( \exp_t \) can achieve the value zero.

Figure 5 shows some \( t \)-exponential functions. They are convex, increasing functions. It is obvious that \( \exp_t \) decays towards 0 more slowly as \( t \) increases. This property leads to a family of heavy-tailed distributions as \( t > 1 \).

Since \( \log_\phi \) is an increasing function, it follows that its inverse \( \exp_\phi \) is also an increasing function. Since \( \phi \) is a non-decreasing function, \( \nabla_x \exp_\phi(x) = (\phi \circ \exp_\phi)(x) \) is also an increasing function. Using Theorem 24.1 in (Rockafellar, 1970) it follows that \( \exp_\phi \) is a strictly convex function.
2.3.3 \( \Phi \)-Exponential Family of Distributions

(Naudts, 2004c) used the \( \phi \)-exponential function to define the parametric distribution family:

\[
p(z; \theta) := \exp_\phi (\langle \Phi(z), \theta \rangle - G_\phi(\theta)).
\]

(13)

where \( \Phi(z) \) is a map from \( z \in \mathbb{Z} \) to the sufficient statistics, and \( \theta \) is the natural parameter. \( G_\phi(\theta) \) is the normalizer of the \( \phi \)-exponential family such that

\[
\int_\mathbb{Z} \exp_\phi (\langle \Phi(z), \theta \rangle - G_\phi(\theta)) \, dz = 1
\]

and \( G_\phi(\theta) \neq \int_\mathbb{Z} \exp_\phi (\Phi(z), \theta) \, dz \) in general. A closely related distribution, which often appears when working with \( \phi \)-exponential families is the so-called escort distribution:

**Definition 2.2 ( Escort distribution)** Let \( \phi : [0, +\infty) \to [0, +\infty) \) be strictly positive and non-decreasing on \( (0, +\infty) \). For a \( \phi \)-exponential family of distributions,

\[
q(z; \theta) := \phi(p(z; \theta))/Z(\theta)
\]

(14)

is called the escort distribution of \( p(z; \theta) \). Here \( Z(\theta) = \int_\mathbb{Z} \phi(p(z; \theta)) \, dz \) is the normalizing constant which ensures that the escort distribution integrates to 1.

One of the crucial properties of exponential families is that the log-partition function \( G \) is convex, and it can be used to generate cumulants of the distribution simply by taking derivatives.

**Theorem 2.3 (Log partition function (Barndorff-Nielsen, 1978))** If the regularity condition

\[
\int_\mathbb{Z} \nabla_\theta p(z; \theta) \, dz = \nabla_\theta \int_\mathbb{Z} p(z; \theta) \, dz
\]

(15)

holds, then

\[
\nabla_\theta G(\theta) = \mathbb{E}[\Phi(z)], \quad \nabla^2_\theta G(\theta) = \text{Var}[\Phi(z)]
\]

(16)

and \( G(\theta) \) is convex.

The proof of the above theorem is included in the Appendix A. Somewhat surprisingly, \( G_\phi(\theta) \) of \( \phi \)-exponential family shares some of the similar properties as \( G(\theta) \) of exponential family. As following theorem asserts, its first derivative can still be written as an expectation of \( \Phi(z) \) but now with respect to the escort distribution in contrast with Theorem 2.3. The proof of the theorem is included in the Appendix B.

**Theorem 2.4 (\( \phi \)-log partition function (Naudts, 2004c; Sears, 2008))** The function \( G_\phi(\theta) \) is convex. Moreover, if the following regularity condition

\[
\int_\mathbb{Z} \nabla_\theta p(z; \theta) \, dz = \nabla_\theta \int_\mathbb{Z} p(z; \theta) \, dz
\]

(17)

holds, then

\[
\nabla_\theta G_\phi(\theta) = \mathbb{E}_{q(z; \theta)}[\Phi(z)]
\]

(18)
Before moving on, we briefly discuss the regularity condition (17), which concerns the legality of swapping the differentiation over a parameter with the integration over the variables. Readers not interested in the following discussion may skip to Section 2.3.4. This is a fairly standard, yet technical requirement, which is often proved using the Dominated Convergence Theorem (see e.g. Section 9.2 of (Rosenthal, 2006)). This holds, for instance, when \( \mathbb{E}_{q(z; \theta)} |\Phi(z)| < \infty \) and \( |\nabla_{\theta} G_{\phi}(\theta)| < \infty \). Here \(| \cdot |\) denotes the \( L_1 \) norm. This condition may not hold for any arbitrary \( \phi \)-exponential family. Here is one example:

**Example 3** Let \( z \in [1, +\infty) \) and \( \Phi(z) = z \). Consider the \( \phi \)-exponential family where \( \phi(x) = x^t \) (later referred as the \( t \)-exponential family), using (11) and (13) the resulting density can be written as

\[
p(z; \theta) = (1 + (1 - t)(\theta z - G_t(\theta)))^{1/(1-t)}.
\]

If we compute

\[
\mathbb{E}_{q(z; \theta)} |\Phi(z)| = \mathbb{E}_{q(z; \theta)} |z| = \frac{1}{Z(\theta)} \int_{1}^{+\infty} (1 + (1 - t)(\theta z - G_t(\theta)))^{t/(1-t)} |z| dz = \frac{1}{Z(\theta)} \int_{1}^{+\infty} \left( z^{(1-t)/t} + (1 - t)(\theta z - G_t(\theta))z^{(1-t)/t} \right)^{t/(1-t)} dz
\]

Whenever \( t \geq 2 \) the integral diverges because

\[
\lim_{z \to +\infty} T_1 + T_2 = O(z^{1/t}), \text{ and hence } \int_{1}^{+\infty} (T_1 + T_2)^{t/(1-t)} dz \to +\infty.
\]

2.3.4 \( t \)-EXPONENTIAL FAMILY OF DISTRIBUTIONS

One of the most important members of the \( \phi \)-exponential family distributions is the \( t \)-exponential family of distributions, which is defined using the \( \exp_t \) function (11) in (13)

\[
p(z; \theta) = \exp_t \left( (\Phi(z), \theta) - G_t(\theta) \right).
\]

(19)

In fact, the \( t \)-exponential family was first proposed in 1980s by Tsallis (Tsallis, 1988; Gell-Mann and Tsallis, 2004)\(^6\). The corresponding escort distribution is given by

\[
q(z; \theta) = \frac{p(z; \theta)^t}{\int p(z; \theta)^t dz}.
\]

(20)

As can be seen in Figure 5, \( \exp_t \), for \( t > 1 \), decays towards 0 more slowly than the \( \exp \) function. Consequently, the \( t \)-exponential family of distributions becomes a family of heavy-tailed distribution as \( t > 1 \).

\(^6\) Note that Tsallis used the term \( q \)-exponential family. However, we prefer using \( t \)-exponential family to avoid confusion between the exponent \( q \) and the escort distribution \( q \).
Although the concept of the $t$-exponential family is relatively new, distributions that belong to this family have been widely used for years. For example, in linear regression problems, it is well-known that the Gaussian distribution is not robust if extreme outliers exist. Instead, the Student’s $t$-distribution is a common substitute in noisy dataset, see e.g. (Zellner, 1976). Interestingly, the Student’s $t$-distribution is actually a member of the $t$-exponential family.

**Example 4 (Student’s-$t$ distribution)** Recall that a $k$-dimensional Student’s-$t$ distribution $St(z | \mu, \Sigma, v)$ with $0 < v < 2$ degrees of freedom has the following probability density function:

$$St(z | \mu, \Sigma, v) = \frac{\Gamma \left( \frac{(v + k)/2}{2} \right)}{\sqrt{(\pi v)^{k/2} \Gamma(v/2) |\Sigma|^{1/2}}} \left( 1 + (z - \mu)^\top (v \Sigma)^{-1} (z - \mu) \right)^{-\frac{(v+k)}/2}. \quad (21)$$

Here $\Gamma(\cdot)$ denotes the usual Gamma function. To see that the Student’s-$t$ distribution is a member of the $t$-exponential family, first set $-(v + k)/2 = 1/(1 - t)$ and

$$\Psi = \left( \frac{\Gamma \left( \frac{(v + k)/2}{2} \right)}{\sqrt{(\pi v)^{k/2} \Gamma(v/2) |\Sigma|^{1/2}}} \right)^{-2/(v+k)}$$

to rewrite (21) as

$$St(z | \mu, \Sigma, v) = \left( \Psi + \Psi \cdot (z - \mu)^\top (v \Sigma)^{-1} (z - \mu) \right)^{1/(1-t)}. \quad (22)$$

Next set $\Phi(z) = [z; z^\top], \theta = [-2\Psi \mathbf{K} \mu/(1 - t); \Psi \mathbf{K} / (1 - t)]$ with $\mathbf{K}$ defined as $\mathbf{K} = (v \Sigma)^{-1}$. Then define

$$\langle \Phi(z), \theta \rangle = \left( \frac{\Psi}{1 - t} \right) (z^\top \mathbf{K} z - 2\mu^\top \mathbf{K} z) \quad \text{and}$$

$$G_t(\theta) = -\left( \frac{\Psi}{1 - t} \right) (\mu^\top \mathbf{K} \mu + 1) + \frac{1}{1 - t}$$

to rewrite (22) as

$$St(z | \mu, \Sigma, v) = (1 + (1 - t) (\langle \Phi(z), \theta \rangle - G_t(\theta)))^{1/(1-t)}.$$

Comparing with (11) clearly shows that

$$St(z | \mu, \Sigma, v) = \exp_t (\langle \Phi(z), \theta \rangle - G_t(\theta)).$$

Using (20) and some simple algebra yields the escort distribution of Student’s-$t$ distribution:

$$q(z; \theta) = St(z | \mu, v \Sigma/(v + 2), v + 2)$$

Interestingly, the mean of the Student’s-$t$ pdf is $\mu$, and its variance is $v \Sigma/(v - 2)$ while the mean and variance of the escort are $\mu$ and $\Sigma$ respectively.
3. Binary Classification

Logistic regression is not robust against random noise, essentially because its conditional distribution is modeled by an exponential family distribution. This section introduces a new algorithm, \(t\)-logistic regression. The motivation of \(t\)-logistic regression follows the same as using the Student’s \(t\)-distribution in linear regression (Zellner, 1976). In classification, we believe that the robustness of logistic regression can also be improved by using a heavy-tailed \(t\)-exponential family distribution.

In \(t\)-logistic regression, we model the conditional likelihood of a data point \((x, y)\) by a \(t\)-exponential family distribution,

\[
p(y | x; \theta) = \exp_t \left( \langle \Phi(x, y), \theta \rangle - G_t(x; \theta) \right)
= \exp_t \left( \frac{n}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right),
\]

where \(t > 1\), and the normalizer \(G_t(x; \theta)\) is the solution of

\[
\exp_t \left( \frac{1}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right) + \exp_t \left( -\frac{1}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right) = 1. \tag{24}
\]

By defining \(\tilde{a} = \langle \Phi(x), \theta \rangle\), \(G_t(\tilde{a}) = G_t(x; \theta)\), we can simplify (24) to,

\[
\exp_t (\tilde{a}^2 - G_t(\tilde{a})) + \exp_t (-\tilde{a}^2 - G_t(\tilde{a})) = 1. \tag{25}
\]

Note that \(G_t(\tilde{a}) = G_t(-\tilde{a})\).

The key challenge in using the \(t\)-exponential family is that no closed form solution\(^7\) exists for computing \(G_t(\tilde{a})\) in (25). However, we provide an iterative method which computes \(G_t(\tilde{a})\) efficiently. The outline of the algorithm is described in Algorithm 1.

**Algorithm 1:** Iterative algorithm for computing \(G_t\) for binary \(t\)-logistic regression.

| Input: \(\tilde{a} \geq 0\) |
| Output: \(G_t(\tilde{a})\) |
| \(\tilde{a} \leftarrow \tilde{a}\); |
| while \(\tilde{a}\) not converged do |
| \(Z(\tilde{a}) \leftarrow 1 + \exp_t (-\tilde{a});\) |
| \(\tilde{a} \leftarrow Z(\tilde{a})^{1-t\tilde{a}};\) |
| end |
| \(G_t(\tilde{a}) \leftarrow -\log_t (1/Z(\tilde{a})) + \frac{\tilde{a}}{2};\) |

The convergence of this iterative algorithm is verified in Appendix G. In practice, the algorithm takes less than 20 iterations to converge to an accuracy of \(10^{-10}\).

We plot the \(t\)-logistic loss

\[
l(x, y, \theta) = -\log \exp_t (\frac{n}{2} - G_t(u))
\]

\(^7\) There are a few exceptions. For example, when \(t = 2\), \(G_t(\tilde{a}) = \sqrt{1 + \frac{\tilde{a}^2}{4}}.\)
as the negative logarithm of (23) as a function of margin \( u = y \hat{a} \) in Figure 6. We find that the \( t \)-logistic loss is quasi-convex and bends down as the margin of a data point becomes too negative. The larger the \( t \), the more is the bending down effect. As \( t = 1 \), the \( t \)-logistic regression reduces to logistic regression, and the loss function becomes convex. This is not surprising since at \( t = 1 \), the \( t \)-exponential family becomes the exponential family.

Mathematically, the bending of the loss is directly related to the gradient of \( t \)-logistic loss function. For a data point \((x, y)\), the gradient with respect to \( \theta \) is,

\[
\nabla_{\theta} l(x, y, \theta) = -\nabla_{\theta} \log \exp_t \left( \frac{y}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right) \\
= -\nabla_{\theta} \left( \frac{y}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right) \exp_t \left( \frac{y}{2} \langle \Phi(x), \theta \rangle - G_t(x; \theta) \right)^{t-1} \\
= -\frac{1}{2} (y \Phi(x) - E_q[y \Phi(x)]) p(y|x; \theta)^{t-1} \\
= -\frac{1}{2} (y - yq(y|x; \theta) + yq(-y|x; \theta)) \Phi(x) p(y|x; \theta) (26) \\
= -yq(-y|x; \theta) \Phi(x) p(y|x; \theta)^{t-1} (28)
\]

where \( q(y|x; \theta) = \frac{p(y|x; \theta)^t}{p(y|x; \theta)^t + p(-y|x; \theta)^t} \), (26) is from (10), and (27) is from (23) and Theorem 2.4.

In (28), the gradient of the loss function of \((x, y)\) is associated with a forgetting variable \( \xi \), which disappears as \( t = 1 \). As \( u = y \langle \Phi(x), \theta \rangle \) gets more negative, \( \xi \) decreases accordingly. Intuitively, the existence of forgetting variable improves the robustness of \( t \)-logistic regression by forgetting the influence of the outliers with low likelihood. We will discuss robustness in more detail in Section 4.2.

4. Properties

In this section, we are going to discuss three key properties of \( t \)-logistic regression. Firstly, we verify Bayes-risk consistency, which is an important statistical property of a binary loss function. Secondly, we formally show that \( t \)-logistic regression is robust against outliers compared to logistic regression. Thirdly, from a practical point of view, we investigate the local minima of non-convex loss functions. We show that the empirical risks of almost all non-convex losses including \( t \)-logistic regression may have multiple local minima. However, in practice, we will show in Section 7 that \( t \)-logistic regression is stable.

4.1 Bayes-Risk Consistency

Since all surrogate losses are a substitute for the \( 0 - 1 \) loss, it is natural to ask whether a surrogate loss is statistical consistent. To answer this question, a crucial criterion which is known as Bayes-risk consistency is used (see e.g. (Bartlett et al., 2006; Masnadi-Shirazi et al., 2010)). Let us denote \( \eta(x) = p(y = 1|x) \) to be the underlying true conditional distribution of the label \( y \) given \( x \), and let \( \hat{a} = \langle \Phi(x), \theta \rangle \). The expected risk of a binary loss function \( l \) is,

\[
C_t(\eta, \hat{a}) = E_y[l(y \hat{a})] = \eta l(\hat{a}) + (1 - \eta) l(-\hat{a}).
\]
Figure 6: $t$-logistic loss for binary classification with four different $t$: $t = 1.3, t = 1.6, t = 1.9$. Unlike the logistic loss (which is $t = 1$), $t$-logistic loss bends down as $u \ll 0$, which caps the influence from outliers.
Since $\text{sign}(\hat{a})$ predicts the label of point based on its feature $x$, Bayes-risk consistency requests the optimal $\hat{a}^*$ of the expected risk $C_l(\eta, \hat{a})$ given $\eta$ to have the same sign as the Bayes decision rule,

$$\text{sign}[\hat{a}^*] = \text{sign}[2\eta - 1]. \quad (29)$$

(Bartlett et al., 2006) further shows that all the three convex surrogate loss functions in Table 1 are Bayes-risk consistent. Now let us verify the Bayes-Risk consistency for $t$-logistic loss

$$l(y\hat{a}) = -\log \exp_t(\frac{y\hat{a}}{2} - G_t(\hat{a})).$$

We have

$$C_l(\eta, \hat{a}) = \eta l(\hat{a}) + (1 - \eta)l(-\hat{a})$$

$$= -\eta \log \exp_t(\frac{\hat{a}}{2} - G_t(\hat{a})) - (1 - \eta) \log \exp_t(-\frac{\hat{a}}{2} - G_t(\hat{a}))$$

$$= -\eta \log \exp_t(\frac{\hat{a}}{2} - G_t(\hat{a})) - (1 - \eta) \log \left(1 - \exp_t(\frac{\hat{a}}{2} - G_t(\hat{a}))\right). \quad (30)$$

where (30) is because of (25). Let us define $r = \exp_t(\frac{\hat{a}}{2} - G_t(\hat{a}))$, and then (30) becomes,

$$-\eta \log(r) - (1 - \eta) \log(1 - r).$$

We can obtain the optimal $r^*$ by taking the derivative of $r$ and set it to 0,

$$\frac{-\eta}{r^*} - \frac{1 - \eta}{1 - r^*} = 0,$$

which yields $r^* = \eta$. Therefore, the optimal $\hat{a}^*$ satisfies $\eta = \exp_t(\frac{\hat{a}^*}{2} - G_t(\hat{a}^*))$. Since $1 - \eta = \exp_t(-\frac{\hat{a}^*}{2} - G_t(\hat{a}^*))$, we can take $\log_t$ of the two and substract them, which yields,

$$\hat{a}^* = \log_t \eta - \log_t(1 - \eta). \quad (31)$$

It is clear that (31) satisfies (29), because $\log_t$ is an increasing function and $\hat{a}^* > 0$ if and only $\eta > \frac{1}{2}$. Therefore, $t$-logistic loss is Bayes-risk consistent.

4.2 Robustness

In this section, we theoretically investigate the robustness of $t$-logistic regression. There is no unique definition of robustness (see e.g. (Hampel et al., 1986; O’hagan, 1979)), and we will mainly focus on two of them. However, both definitions require the computation of the global optimum, which is infeasible for non-convex losses. Instead, we use the necessary conditions of the definitions and propose a function $I_l(u)$ for visualization. Finally, we use $I_l(u)$ to classify binary losses to three robust types, and show that $t$-logistic regression is fundamentally different from convex losses and many other non-convex losses in Table 2.
4.2.1 DEFINITIONS OF ROBUSTNESS

Consider a dataset containing \( m \) data points \( x_1, \ldots, x_m \) with their labels \( y_1, \ldots, y_m \), assume that \( \theta^* \) is the global optimum of the regularized risk

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, \theta) + \frac{\lambda}{2} \|\theta\|_2^2.
\]

For simplicity, let us assume that the loss function \( l(x, y, \theta) \) is continuous and differentiable. From the optimality condition of a differentiable\(^8\) objective function, \( \theta^* \) must satisfy,

\[
\nabla_{\theta} J(\theta^*) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\theta} l(x_i, y_i, \theta^*) + \lambda \theta^* = 0.
\] (32)

Now assume that the dataset is augmented by a contaminated example \((\hat{x}, \hat{y})\). Then the optimum on the contaminated dataset becomes \( \hat{\theta}^* \) and it must satisfy,

\[
\nabla_{\theta} J(\hat{\theta}^*) + \frac{1}{m} \nabla_{\theta} l(\hat{x}, \hat{y}, \hat{\theta}^*) = 0.
\] (33)

The robustness of a loss function is basically determined by the sensitivity of the optimum before and after the addition of a contaminated example, namely the difference between \( \theta^* \) and \( \hat{\theta}^* \).

The two definitions of robustness that we consider are,

**Definition 4.1 (Inspired by influence function in (Hampel et al., 1986))** For any dataset \((x_1, y_1), \ldots, (x_m, y_m)\) and \((\hat{x}, \hat{y})\),

\[
\lim_{m \to \infty} \hat{\theta}^* \to \theta^*.
\]

**Definition 4.2 (Outlier proneness in (O’hagan, 1979))** For any dataset \((x_1, y_1), \ldots, (x_m, y_m)\) and \((\hat{x}, \hat{y})\),

\[
\lim_{\|\Phi(\hat{x})\|_2 \to \infty} \hat{\theta}^* \to \theta^*.
\]

where \( \Phi(x) \) is a feature map from \( \mathcal{X} \) to \( \mathbb{R}^d \).

Roughly speaking, Definition 4.1 states that a robust model should not be affected too much by changing a small portion of data; and Definition 4.2 states that a robust model should ignore any extreme outliers.

However, for non-convex losses, it is very hard to characterize the difference between \( \theta^* \) and \( \hat{\theta}^* \) because the regularized risk may have multiple local minima (see Section 4.3). On the other hand, from (33), it is clear that a necessary condition for \( \hat{\theta}^* \to \theta^* \) is that

\[
\frac{1}{m} \|\nabla_{\theta} l(\hat{x}, \hat{y}, \theta^*)\|_2 \to 0.
\] (34)

Therefore, instead of directly working on Definition 4.1 and 4.2, we will investigate the robustness by their necessary conditions, which are defined in Definition 4.3 and 4.4 respectively.

---

8. For nondifferentiable functions, one can replace the gradient by the subgradient and obtain a similar optimality condition.
Definition 4.3 For any $\mathbf{x}$, $y$ and $\mathbf{\theta}$,
$$\|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 < \infty.$$

Definition 4.4 For any $\mathbf{x}$, $y$ and $\mathbf{\theta}$,
$$\lim_{\|\Phi(\mathbf{x})\|_2 \to \infty} \|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 = 0.$$

4.2.2 Robust Types
Since $\|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2$ involves both $\mathbf{\theta}$ and $(\mathbf{x}, y)$, it would be more straightforward to have a function of margin $u = y \langle \Phi(\mathbf{x}), \mathbf{\theta} \rangle$ for visualization. To this end, we do an inner-product between $\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})$ and $\mathbf{\theta}$, and define a new function $I_l(u)$,
$$\langle \nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta}), \mathbf{\theta} \rangle = \langle l'(u) y \Phi(\mathbf{x}), \mathbf{\theta} \rangle = l'(u) u := I_l(u),$$
where $l(u) := l(\mathbf{x}, y; \mathbf{\theta})$ and $l'(u) \leq 0$ for all losses that we are interested in this paper.

Furthermore, the following two lemmas show that $I_l(u)$ and $\|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2$ can almost equivalently define robustness. The proofs of the lemmas are provided in Appendix C and D.

Lemma 4.5 If $|I_l(u)| < \infty$, then for any $\mathbf{\theta}$, $\mathbf{x}$ and $y$, the probability
$$p(\|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 < \infty) = 1.$$  
Furthermore, $\|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 \to \infty$ if and only if $\psi$ the angle between $\Phi(\mathbf{x})$ and $\mathbf{\theta}$ is equal to $\pi/2$ and $\|\Phi(\mathbf{x})\|_2 \to \infty$.

Lemma 4.6 If $\lim_{u \to \infty} |I_l(u)| = 0$, then for any $\mathbf{\theta}$, $\mathbf{x}$ and $y$, the probability
$$p(\lim_{\|\Phi(\mathbf{x})\|_2 \to \infty} \|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 = 0) = 1.$$  
Furthermore, $\lim_{\|\Phi(\mathbf{x})\|_2 \to \infty} \|\nabla_{\mathbf{\theta}} l(\mathbf{x}, y, \mathbf{\theta})\|_2 \neq 0$ if and only if $\psi$ the angle between $\Phi(\mathbf{x})$ and $\mathbf{\theta}$ is equal to $\pi/2$.

Since all the losses that are commonly used are continuously defined on $u \in \mathbb{R} \cup \{\pm \infty\}$, we have $|l'(u)| < \infty$ for $|u| < \infty$. Therefore, $|I_l(u)|$ may only be unbounded as $u \to \infty$. Based on $\lim_{u \to \infty} |I_l(u)|$, we classify binary losses into three robust types:

- Robust Loss 0: $\lim_{u \to \infty} |I_l(u)| \to \infty$.
- Robust Loss I: $0 < \lim_{u \to \infty} |I_l(u)| < \infty$.
- Robust Loss II: $\lim_{u \to \infty} |I_l(u)| = 0$.

An illustration of the three types of binary losses is provided in Figure 7. In Table 3, we classify some binary losses based on their robustness. It is easy to verify that all convex losses belongs to Robust Loss 0. Some other verifications are provided in Appendix H. In particular, it differentiates the $t$-logistic regression (Type-I) from Type-0 losses, e.g. logistic regression as well as the Type-II non-convex losses, e.g. Savage loss. In later experiments, we will empirically compare these different types of losses.
Figure 7: An illustration of the three robust types. All the three types of losses behave similarly as \( u > 0 \). When \( u \to -\infty \), Type-0 loss goes to \( +\infty \); Type-I loss goes to a constant; and Type-II loss goes to 0.

Table 3: The robustness of some loss functions for binary classification based on \( I_l(u) \). The verifications are provided in Appendix H.

<table>
<thead>
<tr>
<th>Name</th>
<th>Loss Function</th>
<th>Robust Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hinge</td>
<td>( l(x, y, \theta) = \max(0, 1-u) )</td>
<td>0</td>
</tr>
<tr>
<td>Exponential</td>
<td>( l(x, y, \theta) = \exp(-u) )</td>
<td>0</td>
</tr>
<tr>
<td>Logistic</td>
<td>( l(x, y, \theta) = \log(1 + \exp(-u)) )</td>
<td>0</td>
</tr>
<tr>
<td>T-logistic</td>
<td>( l(x, y, \theta) = \exp_t\left(\frac{u}{2} - G_t(u)\right) )</td>
<td>I</td>
</tr>
<tr>
<td>Probit</td>
<td>( l(x, y, \theta) = 1 - \text{erf}(u) )</td>
<td>II</td>
</tr>
<tr>
<td>Ramp</td>
<td>( l(x, y, \theta) = \min(2, \max(0, 1-u)) )</td>
<td>II</td>
</tr>
<tr>
<td>Savage</td>
<td>( l(x, y, \theta) = \frac{4}{1+\exp(u)} )</td>
<td>II</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>( l(x, y, \theta) = \frac{2}{1+\exp(u)} )</td>
<td>II</td>
</tr>
</tbody>
</table>

4.3 Multiple Local Minima

One of the key disadvantages of the non-convex losses is that its empirical risk may have multiple local minima. To illustrate this, we used a two-dimensional toy dataset which contains 50 points drawn uniformly from \([-2, 2] \times [-2, 2]\). In comparison, we plot the empirical risk of \( t \)-logistic loss as well as Savage loss in Figure 8.

As can be seen, Savage loss yields a highly non-convex objective function with a large number of local optima. In contrast, even though we are averaging over non-convex loss functions, the...
Figure 8: The empirical risk of $t$-logistic regression (upper) and Savage loss (lower) on a toy two dimensional dataset. $T$-logistic regression appears to be easier to optimize than Savage loss.
resulting function of $t$-logistic regression has a single global optimum. This behavior persists when we use different random samples, change the sampling scheme, or vary the number of data points. Moving over to higher dimensional datasets such as Adult, USPS, and Web8\(^9\), we initialize the algorithm with different randomly chosen starting points and check the solution obtained. The algorithm always arrives at the same solution (within numerical precision) (Ding and Vishwanathan, 2010).

This interesting behavior once led us to conjecture that $t$-logistic regression has only one local minimum which is the global minimum. However, in the next theorem, we show that the conjecture is wrong. For any non-convex loss function, under some mild conditions, one can always construct a dataset whose empirical risk has multiple local minima. To the best of our knowledge, all the existing non-convex losses satisfy the conditions. The following theorem considers the case when the feature dimension is 1-dimensional and generalization to multi-dimensional setting is straightforward. The proof of the theorem is in the Appendix E.

**Theorem 4.7** Consider a loss function $l(u) := l(x,y,\theta)$ that is smooth at $u := y\theta x = 0$. If $l'(0) < 0$, and there exist $u_1 < 0$, $u_2 > 0$, and $\epsilon > 0$ where $l'(u_i) \geq l'(0) + \epsilon$ for $i = 1, 2$, then there exists a dataset whose empirical risk $R_{\text{emp}}(\theta)$ has at least two local minima.

An interesting observation is that those local minima are related to the robustness of non-convex losses. To see that, consider the following 1-dimensional example, which includes 30 clean data points where $(x_i, y_i) = (1, 1)$, and one outlier where $(x, y) = (-200, 1)$. We plot the empirical risk as a function of $\theta$ for both the logistic regression and $t$-logistic regression in Figure 9. Without the outlier being added, both logistic loss (red dashed) and $t$-logistic loss (purple solid) yield the same optimum. However, once the outlier is added, the optimum of the logistic regression is severely impacted by the outlier (blue dashed). On the other hand, $t$-logistic regression, although it creates another local minimum, retains the same global optimum as the $\theta^*$ without the outlier (green solid).

5. Multiclass Classification

In this section, we extend $t$-logistic regression to multiclass classification, where the dataset consists of data points $\{x_1, \ldots, x_m\}$ and the corresponding label $\{y_1, \ldots, y_m\}$ with $y_i$ taking values from $\{1, \ldots, C\}$. Let us first briefly review the multiclass logistic regression. The generalization to the $t$-logistic regression is straightforward.

In the multiclass logistic regression, the conditional likelihood of a label $y$ given $x$ is

$$p(y|\mathbf{x}; \mathbf{\theta}) = \exp\left(\langle \Phi(\mathbf{x}, y), \mathbf{\theta} \rangle - G(\mathbf{x}; \mathbf{\theta}) \right)$$

where,

$$\Phi(\mathbf{x}, y) = \begin{pmatrix} 0, \ldots, 0, \Phi(\mathbf{x}), 0, \ldots, 0 \\ 1, \ldots, y-1, y+1, \ldots, C \end{pmatrix}, \ \mathbf{\theta} = (\theta_1, \ldots, \theta_C)$$

with $\Phi(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^d$. Here $\mathbf{0}$ denotes the $d$-dimensional all-zero vector, and $\mathbf{\theta}$ a $d \times C$-dimensional vector.

Figure 9: Empirical risk of logistic regression and $t$-logistic regression on the one dimensional example. The optimal solutions before and after adding the outlier are significantly different for logistic regression. In contrast, the global optimum of $t$-logistic regression stays the same.
Therefore,

\[ p(y|\mathbf{x}; \theta) = \exp \left( \langle \Phi(\mathbf{x}), \theta_y \rangle - G(\mathbf{x}; \theta) \right), \tag{35} \]

where the log-partition function is

\[ G(\mathbf{x}; \theta) = \log \left( \sum_{c=1}^{C} \exp \left( \langle \Phi(\mathbf{x}), \theta_c \rangle \right) \right). \tag{36} \]

The multiclass logistic loss is the negative log-likelihood of a point \((\mathbf{x}, y)\), which equals

\[ l(\mathbf{x}, y, \theta) = -\log p(y|\mathbf{x}; \theta) = \log \left( \sum_{c=1}^{C} \exp \left( \langle \Phi(\mathbf{x}), \theta_c - \theta_y \rangle \right) \right). \]

The main idea of the multiclass \(t\)-logistic regression is the same as in binary \(t\)-logistic regression. The conditional likelihood of the data point \((\mathbf{x}, y)\) is modeled by a conditional \(t\)-exponential family of distributions \((t > 1)\):

\[ p(y|\mathbf{x}, \theta) = \exp_t \left( \langle \Phi(\mathbf{x}, y), \theta \rangle - G_t(\mathbf{x}; \theta) \right) \]
\[ = \exp_t \left( \langle \Phi(\mathbf{x}), \theta_y \rangle - G_t(\mathbf{x}; \theta) \right) \tag{37} \]

where the log-partition function \(G_t\) satisfies

\[ \sum_{c=1}^{C} \exp_t \left( \langle \Phi(\mathbf{x}), \theta_c \rangle - G_t(\mathbf{x}; \theta) \right) = 1. \tag{38} \]

Let \(\mathbf{a}_c = \langle \Phi(\mathbf{x}), \theta_c \rangle\), \(G_t(\mathbf{a}) = G_t(\mathbf{x}; \theta)\), then we can simplify (38) as

\[ \sum_{c=1}^{C} \exp_t(\mathbf{a}_c - G_t(\mathbf{a})) = 1. \]

An iterative algorithm which generalizes the one used in binary classification is applied to compute \(G_t(\mathbf{a})\). The algorithm is described in Algorithm 2.

**Algorithm 2**: Iterative algorithm for computing \(G_t\) in multiclass \(t\)-logistic regression.

**Input**: \(\mathbf{a}\)

**Output**: \(G_t(\mathbf{a})\)

\[
\mathbf{a}^* \leftarrow \text{max}(\mathbf{a}); \\
\mathbf{\hat{a}} \leftarrow \mathbf{a} - \mathbf{a}^*; \\
\text{while } \mathbf{\hat{a}} \text{ not converged do} \\
\quad Z(\mathbf{\hat{a}}) \leftarrow \sum_{c=1}^{C} \exp_t(\mathbf{\hat{a}}_c); \\
\quad \mathbf{\hat{a}} \leftarrow Z(\mathbf{\hat{a}})^{1-t}(\mathbf{\hat{a}} - \mathbf{\hat{a}}^*); \\
\text{end} \\
G_t(\mathbf{\hat{a}}) \leftarrow -\log_t(1/Z(\mathbf{\hat{a}})) + \mathbf{\hat{a}}^*; 
\]
Table 4: Average time (in milliseconds) spent by our iterative scheme and \texttt{fsolve} in computing $\hat{G}_t(\hat{a})$ for multiclass $t$-logistic regression.

<table>
<thead>
<tr>
<th>C</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{fsolve}</td>
<td>8.1</td>
<td>8.3</td>
<td>8.1</td>
<td>8.7</td>
<td>9.6</td>
<td>9.8</td>
<td>10.0</td>
<td>10.2</td>
<td>10.3</td>
<td>10.7</td>
</tr>
<tr>
<td>iterative</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
</tr>
</tbody>
</table>

In practice, Algorithm 2 scales well with $C$, the number of classes, thus making it efficient enough for problems involving a large number of classes. To illustrate, we let $C \in \{10, 20, \ldots, 100\}$ and we randomly generate $\hat{a} \in [-10, 10]^C$, and compute the corresponding $\hat{G}_t(\hat{a})$. We compare the time spent in estimating $\hat{G}_t(\hat{a})$ by the iterative scheme and by calling Matlab \texttt{fsolve} function averaged over 100 random generations using Matlab 7.1 in a 2.93 GHz Dual-Core CPU. We present the results in Table 4.

For a data point $(x, y)$, the partial derivative of multiclass $t$-logistic loss function over $\theta_n$, where $n \in \{1, \ldots, C\}$, is

$$\frac{-\partial}{\partial \theta_n} \log p(y \mid x; \theta) = \frac{-\partial}{\partial \theta_n} \log \exp_t(\langle \Phi(x), \theta_y \rangle - G_t(x; \theta))$$

$$= - (\delta(y = n)\Phi(x) - \mathbb{E}_q[\Phi(x, y)])p(y \mid x; \theta)^t-1$$

$$= - \Phi(x) \cdot (\delta(y = n) - \sum_{c=1}^C \delta(c = n)q(c \mid x; \theta))p(y \mid x; \theta)^t-1$$

$$= - \Phi(x) \cdot (\delta(y = n) - q(n \mid x; \theta)\xi)$$

(39)

where $q(n \mid x; \theta) = \frac{p(n \mid x; \theta)^t}{\sum_{c=1}^C p(c \mid x; \theta)^t}$. In (39), the gradient of $(x, y)$ contains a forgetting variable $\xi = p(y \mid x; \theta)^t-1$. Just like the binary classification, when $t > 1$, the influence of the points with low likelihood $p(y \mid x; \theta)$ will be capped by the $\xi$ variable.

The definition of Bayes-risk consistency of multiclass classification losses was first discussed in (Tewari and Bartlett, 2007). As one can easily verify, multiclass $t$-logistic regression is also Bayes-risk consistent (see Appendix I for verification).

6. Optimization Methods

In this section, let us consider some of the practical issues including how to optimize the objective function of $t$-logistic regression. The most straightforward way is to use a gradient-based method such as L-BFGS. In particular, the gradient of $t$-logistic regression is given in (28) for binary classification and (39) for multiclass classification.

Although in our experiment we find that the algorithm converges every time using the L-BFGS solver, it is important to note that there is no convergence guarantee for using L-BFGS solver on non-convex objective functions. In the remainder of the section, we provide a different approach which is guaranteed to converge. For clarity, we discuss how to optimize its empirical risk. The regularized risk can be optimized in a similar way, which was applied in (Ding and Vishwanathan, 2010).
\section{Convex Multiplicative Programming}

For $t > 1$, instead of directly minimizing $R_{\text{emp}}(\theta) = -\log p(y \mid X, \theta)$, one can equivalently minimize the objective function $p(y \mid X, \theta)^{1-t}$,

\begin{equation}
P(\theta) \triangleq p(y \mid X; \theta)^{1-t} = \prod_{i=1}^{m} p(y_i \mid x_i; \theta)^{1-t} \tag{40}
\end{equation}

\begin{equation}
= \prod_{i=1}^{m} \left( 1 + (1-t) \left( \langle \Phi(x_i, y_i), \theta \rangle - G_t(x_i; \theta) \right) \right) / l_i(\theta) \tag{41}
\end{equation}

Since $t > 1$, and $G_t(x_i; \theta)$ is convex, it is easy to see that each component $l_i(\theta)$ is positive and convex. Therefore, $P(\theta)$ becomes the product of positive convex functions $l_i(\theta)$. Minimizing such a function $P(\theta)$ is also called convex multiplicative programming \cite{Kuno1993}.

The optimal solutions to the problem (41) can be obtained by solving the following parametric problem (see Theorem 2.1 of \cite{Kuno1993}):

\begin{equation}
\min_{\zeta} \min_{\theta} \text{MP}(\theta, \zeta) \triangleq \sum_{i=1}^{m} \zeta_i l_i(\theta) \quad \text{s.t.} \quad \zeta > 0, \quad \prod_{i=1}^{m} \zeta_i \geq 1. \tag{42}
\end{equation}

Exact algorithms have been proposed for solving (42) (for instance, \cite{Kuno1993}). However, the computational cost of these algorithms grows exponentially with respect to $m$, which makes them impractical for our purposes. Instead, we apply the following block coordinate descent based method, namely the $\zeta$-$\theta$ algorithm. The main idea of the algorithm is to minimize (42) with respect to $\theta$ and $\zeta$ separately.

\textbf{$\zeta$-Step:} Assume that $\theta$ is fixed, and denote $\tilde{l}_i = l_i(\theta)$ to rewrite (42) as:

\begin{equation}
\min_{\zeta} \text{MP}(\theta, \zeta) = \min_{\zeta} \sum_{i=1}^{m} \zeta_i \tilde{l}_i \quad \text{s.t.} \quad \zeta > 0, \quad \prod_{i=1}^{m} \zeta_i \geq 1. \tag{43}
\end{equation}

Since the objective function is linear in $\zeta$ and the feasible region is a convex set, (43) is a convex optimization problem. By introducing a non-negative Lagrange multiplier $\gamma \geq 0$, the Lagrangian and its partial derivative with respect to $\zeta_i'$ can be written as

\begin{equation}
L(\zeta, \gamma) = \sum_{i=1}^{m} \zeta_i \tilde{l}_i + \gamma \cdot \left( 1 - \prod_{i=1}^{m} \zeta_i \right) \tag{44}
\end{equation}

\begin{equation}
\frac{\partial}{\partial \zeta_i'} L(\zeta, \gamma) = \tilde{l}_i' - \gamma \prod_{i \neq i'} \zeta_i. \tag{45}
\end{equation}

Setting the gradient to 0 obtains $\gamma = \prod_{i \neq i'} \zeta_i$. Since $\tilde{l}_i' > 0$, it follows that $\gamma$ cannot be 0. By the K.K.T. conditions (\cite{Boyd2004}), $\prod_{i=1}^{m} \zeta_i = 1$. This in turn implies that $\gamma = \tilde{l}_i' \zeta_i'$ or

\begin{equation}
(\zeta_1, \ldots, \zeta_m) = (\gamma/\tilde{l}_1, \ldots, \gamma/\tilde{l}_m), \quad \text{with} \quad \gamma = \prod_{i=1}^{m} \tilde{l}_i^{\gamma}. \tag{46}
\end{equation}
There is an obvious connection between $\zeta_i$ and the forgetting variable $\xi_i$, because $\zeta_i \propto 1/\tilde{l}_i = p(y_i | x_i, \theta)^{t-1} = \xi_i$.

**θ-Step**: In this step we fix $\zeta$ and solve for the optimal $\theta$. This step is essentially the same as logistic regression, except that each component has a weight $\zeta_i$ here.

$$\min_{\theta} \text{MP}(\theta, \zeta) = \min_{\theta} \sum_{i=1}^{m} \zeta_i l_i(\theta)$$

and the gradient is

$$\nabla_\theta \text{MP}(\theta, \zeta) = (1 - t) \sum_{i=1}^{m} \zeta_i (\Phi(x_i, y_i) - \mathbb{E}_q[\Phi(x_i, y_i)])$$

The gradient in (48) is very similar to the gradient in (28), (39). The main difference is that the $\zeta$-$\theta$ algorithm computes $\zeta$ and $\theta$ in two steps, while the gradient based method computes $\xi$ and $\theta$ in one step. However, the advantage of the $\zeta$-$\theta$ algorithm is its convergence guarantee as shown in the following theorem. The proof is provided in the Appendix F.

**Theorem 6.1** The $\zeta$-$\theta$ algorithm converges to a stationary point of the convex multiplicative programming problem.

### 7. Empirical Evaluation

We used 26 publicly available binary classification datasets and 9 multiclass classification datasets and focused our study on two aspects: the generalization performance under various noise models and the stability of the solution under random initialization. As our comparator we use logistic regression, and Savage loss.

Our main observation from these extensive empirical experiments is that $t$-logistic regression is more robust than logistic regression, when the dataset is mixed with label noise. On the other hand, compared to Savage loss which often gets stuck in different local minima under random initializations, $t$-logistic regression appears to be much more stable. These two observations make $t$-logistic regression an attractive algorithm for classification.

**Datasets**  Table 5 summarizes the binary classification datasets used in our experiments. adult9, astro-ph, news20, real-sim, reuters-cl11, reuters-ccat are from the same source as in Hsieh et al. (2008). aut-avn is from Andrew McCallum’s home page, covertype is from the UCI repository (Merz and Murphy, 1998), worm is from Franc and Sonnenburg (2008), kdd99 is from KDD Cup 1999, while web8, webspam-u, webspam-t, as well as the kdda

10. The multiclass Savage loss is defined as,

$$l(x, y; \theta) = \sum_{c=1}^{C} \left( \delta(y = c) - \frac{\exp((\Phi(x), \theta_c))}{\sum_{c=1}^{C} \exp((\Phi(x), \theta_c))} \right)^2$$

13. webspam-u is the webspam-unigram and webspam-t is the webspam-trigram dataset. Original dataset can be found at http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html.
Table 5: Summary of the binary classification datasets used in our experiments. \( n \) is the total # of examples, \( d \) is the # of features, and \( n_+:n_- \) is the ratio of the number of positive vs negative examples. M denotes a million. * deontes the datasets that are unused in Chapter 6.

<table>
<thead>
<tr>
<th>dataset</th>
<th>( n )</th>
<th>( d )</th>
<th>( n_+:n_- )</th>
<th>dataset</th>
<th>( n )</th>
<th>( d )</th>
<th>( n_+:n_- )</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult9</td>
<td>48,842</td>
<td>123</td>
<td>0.32</td>
<td>alpha</td>
<td>500,000</td>
<td>500</td>
<td>1.00</td>
</tr>
<tr>
<td>astro-ph</td>
<td>99,757</td>
<td>123</td>
<td>0.31</td>
<td>aut-avn</td>
<td>20,707</td>
<td>54</td>
<td>0.57</td>
</tr>
<tr>
<td>beta</td>
<td>500,000</td>
<td>100</td>
<td>1.00</td>
<td>covertype</td>
<td>581,012</td>
<td>54</td>
<td>1.07</td>
</tr>
<tr>
<td>delta</td>
<td>500,000</td>
<td>500</td>
<td>1.00</td>
<td>dna*</td>
<td>800.4M</td>
<td>500</td>
<td>1.00</td>
</tr>
<tr>
<td>epsilon*</td>
<td>500,000</td>
<td>2000</td>
<td>1.00</td>
<td>gamma</td>
<td>500,000</td>
<td>500</td>
<td>1.00</td>
</tr>
<tr>
<td>kdd99</td>
<td>5.21 M</td>
<td>127</td>
<td>4.04</td>
<td>kdda*</td>
<td>8.92 M</td>
<td>20.22 M</td>
<td>5.80</td>
</tr>
<tr>
<td>kddb*</td>
<td>20.01 M</td>
<td>29.89 M</td>
<td>6.18</td>
<td>longservedio</td>
<td>2000</td>
<td>21</td>
<td>1.00</td>
</tr>
<tr>
<td>measewyner</td>
<td>2000</td>
<td>20</td>
<td>1.00</td>
<td>mushrooms</td>
<td>8124</td>
<td>112</td>
<td>1.07</td>
</tr>
<tr>
<td>news20</td>
<td>19,954</td>
<td>7.26 M</td>
<td>1.00</td>
<td>ocr*</td>
<td>3.50 M</td>
<td>1156</td>
<td>0.96</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,201</td>
<td>2.97 M</td>
<td>0.44</td>
<td>reuters-c11</td>
<td>804,414</td>
<td>1.76 M</td>
<td>0.03</td>
</tr>
<tr>
<td>reuters-ccat</td>
<td>804,414</td>
<td>1.76 M</td>
<td>0.90</td>
<td>web8</td>
<td>59,245</td>
<td>300</td>
<td>0.03</td>
</tr>
<tr>
<td>webspam-t*</td>
<td>350,000</td>
<td>16.61 M</td>
<td>1.54</td>
<td>webspam-u</td>
<td>350,000</td>
<td>254</td>
<td>1.54</td>
</tr>
<tr>
<td>worm</td>
<td>1.03 M</td>
<td>804</td>
<td>0.06</td>
<td>zeta</td>
<td>500,000</td>
<td>800.4 M</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 6: Summary of the multiclass classification datasets used in our experiments. \( n \) is the total # of examples, \( d \) is the # of features, \( nc \) is the # of classes. * deontes the datasets that are unused in Chapter 6.

<table>
<thead>
<tr>
<th>dataset</th>
<th>( n )</th>
<th>( d )</th>
<th>( nc )</th>
<th>dataset</th>
<th>( n )</th>
<th>( d )</th>
<th>( nc )</th>
</tr>
</thead>
<tbody>
<tr>
<td>dna</td>
<td>2,586</td>
<td>182</td>
<td>3</td>
<td>letter</td>
<td>15,500</td>
<td>18</td>
<td>26</td>
</tr>
<tr>
<td>mnist</td>
<td>70,000</td>
<td>782</td>
<td>10</td>
<td>protein</td>
<td>21,516</td>
<td>359</td>
<td>3</td>
</tr>
<tr>
<td>rcv1*</td>
<td>534,130</td>
<td>47,238</td>
<td>52</td>
<td>sensitacoustic</td>
<td>98,528</td>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>sensitcombined</td>
<td>98,528</td>
<td>102</td>
<td>3</td>
<td>sensitseismic</td>
<td>98,528</td>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>usps</td>
<td>9298</td>
<td>258</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and kddb\(^{14}\) are from the LibSVM binary data collection\(^{15}\). The alpha, beta, delta, dna, epsilon, gamma, ocr and zeta datasets were obtained from the Pascal Large Scale Learning Workshop website (Sonnenburg et al., 2008). measewyner is a synthetic dataset proposed in Mease and Wyner (2008). The input \( x \) is a 20-dimensional vector where each coordinate is uniformly distributed on \([0, 1]\). The label \( y \) is \( +1 \) if \( \sum_{j=1}^{10} x_j \geq 2.5 \) and \( -1 \) otherwise.

Table 6 summarizes the multiclass classification datasets. In dna and ocr binary classification datasets, we used the same training and testing partition as in (Zhang et al., 2013) (80% for training and 20% for testing). For all other datasets, we used 70% of the labeled data for training and the remaining 30% for testing. In all cases, we added a constant feature as bias.

\(^{14}\) These datasets were derived from KDD CUP 2010. kdda is the first problem algebra,2008,2009 and kddb is the second problem bridge_to_algebra,2008,2009.
Optimization algorithms We choose to optimize the empirical risk with $L_2$ regularizer using the L-BFGS. We implemented all the loss functions using PETSc and TAO, which allows efficient use of large-scale parallel linear algebra. We used the Limited Memory Variable Metric (lmvm) variant of L-BFGS which is implemented in TAO. The convergence criterion of the optimization algorithms is when the decrease in the objective function value and the norm of the gradient are less than $10^{-10}$ or the maximum number of 1000 function evaluations is reached.

Implementation and Hardware All experiments are conducted on the Rossmann computing cluster at Purdue University, where each node has two 2.1GHz 12-core AMD 6172 processors with 48 Gb physical memory. We ran our algorithms with 4 cores in one single node for all datasets, except dna and ocr datasets for binary classification, where we used 16 cores across 16 nodes with 30 Gb memory in each node.

7.1 Noise Models

One of the main objectives of our experiment is to test the robustness of the classification algorithms under different label noise models. Therefore, we implement the following three kinds of noise models. For binary classification, the three types of noise models are generated in the following ways using a flipping constant $\rho \in [0, 1]$:

**Uniform Noise (Noise-1)** In noise-1 model, we uniformly flip the label of a training example with probability $\rho$ (See Algorithm 3).

**Unbalanced Noise (Noise-2)** Noise-2 model generates unbalanced label noise. In other words, we only flip the labels of negative-label training examples with probability $\rho$ (See Algorithm 4).

**Unbalanced Large-Margin Noise (Noise-3)** Noise-3 model is intended to generate large-margin outliers. In addition, only examples with the negative labels will be flipped. In order to estimate the margin of each example, we first run logistic regression on the clean dataset. We then flip the labels of examples by using the probability which favors the large margin examples (See Algorithm 5).

For multiclass classification, in noise-1 model, we assign $y_i$ to a uniformly random new label with probability $\rho$. In noise-2 and noise-3 model, we only change the labels of the examples with $y_i \leq C/2$, where $C$ is the total number of classes. The new assigned label $y_i$ will be in $(C/2, C]$ based on uniform distribution.
Algorithm 4: Algorithm for generating noise-2 model.

\textbf{Input}: Dataset \((X, Y) := \{(x_i, y_i)\}, \text{where } i = 1, \ldots, m.\)

\textbf{Output}: Dataset \((\tilde{X}, \tilde{Y}) := \{(\tilde{x}_i, \tilde{y}_i)\}, \text{where } i = 1, \ldots, m.\)

\begin{algorithm}
\begin{algorithmic}
\For {\(i = 1, \ldots, m\)}
\State \(\text{rand} = \text{Uni}[0, 1];\)
\If {\(y_i < 0 \text{ and } \text{rand} < \rho\)}
\State \(\tilde{y}_i = 1;\)
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

Algorithm 5: Algorithm for generating noise-3 model.

\textbf{Input}: Dataset \((X, Y) := \{(x_i, y_i)\}, \text{where } i = 1, \ldots, m.\)

\textbf{Output}: Dataset \((\tilde{X}, \tilde{Y}) := \{(\tilde{x}_i, \tilde{y}_i)\}, \text{where } i = 1, \ldots, m.\)

Train \(\theta\) by running logistic regression on \((X, Y)\) for 30 iterations;

\begin{algorithm}
\begin{algorithmic}
\For {\(i = 1, \ldots, m\)}
\State Compute \(u_i = y_i \langle \Phi(x_i), \theta \rangle;\)
\EndFor
\State Compute \(u_{\max} = \max_i \{u_i\};\)
\For {\(i = 1, \ldots, m\)}
\State Compute \(\tilde{u}_i = u_i - u_{\max};\)
\EndFor
\State Compute \(\tilde{u}_{\min} = \min\{\tilde{u}_i\};\)
\For {\(i = 1, \ldots, m\)}
\State Compute \(b_i = \exp(-\frac{10 \cdot \tilde{u}_i}{\tilde{u}_{\min}});\)
\EndFor
\State Compute \(Z = \sum_i \exp(-\frac{10 \cdot \tilde{u}_i}{\tilde{u}_{\min}});\)
\For {\(i = 1, \ldots, m\)}
\State \(\text{rand} = \text{Uni}[0, 1];\)
\If {\(y_i < 0 \text{ and } \text{rand} \cdot Z < m \cdot b_i \cdot \rho\)}
\State \(\tilde{y}_i = 1;\)
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}
7.2 Experiment Design

Since most of our datasets contain a large amount of features, we used the identity feature map \( \Phi(x) = x \) throughout the experiment. For \( t \)-logistic regression, we set \( t = 1.5 \).

7.2.1 Generalization Performance

Our first experiment is to compare the test error among the three algorithms under three different noise models with \( \rho = \{0.00, 0.05, 0.10\} \). We split the training set into 5 partitions for 5-fold cross validation. The candidates of regularization constant \( \lambda \) are \( \{10^{-2}, 10^{-4}, \ldots, 10^{-10}\} \), and the one which in average performs best in the validation sets is chosen. The model parameters in this experiment are initialized to be all zero.

7.2.2 Random Initialization

The second experiment is intended to compare the stability of non-convex losses. In particular, we want to test whether the non-convex losses get stuck in different local minima when we initialize the model parameters differently. We use the regularization constant chosen with 5-fold cross validation in the previous experiment and pick one of the five folds for training. In order to obtain random initialization of the model parameters, each variable of the model parameter is initialized uniformly randomly from \([-10, 10]\). The mean and the standard deviation of the test error is reported from nine random initializations and one all-zero initialization.

For \( \text{dna} \) and \( \text{ocr} \) dataset, due to the large computational cost, we only report the generalization performance with all-zero initialization. We do not split the training set for cross validation, but train the algorithm on the entire training set with \( \lambda = 10^{-10} \).

7.3 Results

From Figure ?? to Figure ??, we plot the performance of the three algorithms under three noise models from left to right. Each figure is the performance on one dataset. For each noise model, we report the test error of the algorithms with \( \rho = 0.00 \) (blue), 0.05 (red), 0.10 (yellow).

On the first row of each figure, we report the test error of the three algorithms using 5-fold cross validation with the optimal \( \lambda \) on that dataset. For large values of \( \lambda \) (e.g. \( \lambda = \{10^{-2}, 10^{-4}\} \)), it appears that the test performance of the algorithms are mostly inferior. This is because most of the datasets we use in our experiment contain a large number of examples, and therefore requires very mild regularization. On the other hand, the dataset with higher noise tends to require larger regularization. For instance, for the binary classification, if \( \rho = 0.00 \), the distribution for the optimal \( \lambda \) equal to \( [10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}] \) is \( [7, 18, 90, 43, 58] \), while the distribution becomes \([16, 36, 82, 36, 46]\) if \( \rho = 0.10 \). To quickly overview the improvement of robustness for \( t \)-logistic regression, in Table 7 and Table 8, we summarize the number of datasets where the test error difference between logistic regression and \( t \)-logistic regression are significant in three noise models with \( \rho = \{0.00, 0.05, 0.10\} \). Across a variety of binary classification datasets in Table 7, it appears that \( t \)-logistic regression performs better when label noise is added. In particular, when \( \rho = 0.05 \), \( t \)-logistic regression has significant advantage in 48 cases, while logistic regression only has 12 cases. When \( \rho = 0.10 \), the advantage of \( t \)-logistic reduces, but still it is better in 42 cases versus 12 for logistic regression. In multiclass classification as shown in Table 8, the advantage of \( t \)-logistic regression is even more salient.
Table 7: The number of binary classification datasets that logistic regression or \( t \)-logistic regression is significantly better than the other. The left part of the table is the number of datasets where logistic regression is significantly better; the right part is the number of datasets where \( t \)-logistic regression is significantly better. The total number of datasets is 24 (dna and ocr datasets are excluded).

<table>
<thead>
<tr>
<th></th>
<th>Logistic</th>
<th>0.00</th>
<th>0.05</th>
<th>0.10</th>
<th>( t = 1.5 )</th>
<th>0.00</th>
<th>0.05</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise-1</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td></td>
<td>Noise-1</td>
<td>5</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>Noise-2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td></td>
<td>Noise-2</td>
<td>5</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Noise-3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td>Noise-3</td>
<td>5</td>
<td>17</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 8: The number of multiclass classification datasets that logistic regression or \( t \)-logistic regression is significantly better than the other. The left part of the table is the number of datasets where logistic regression is significantly better; the right part is the number of datasets where \( t \)-logistic regression is significantly better. The total number of datasets is 9.

<table>
<thead>
<tr>
<th></th>
<th>Logistic</th>
<th>0.00</th>
<th>0.05</th>
<th>0.10</th>
<th>( t = 1.5 )</th>
<th>0.00</th>
<th>0.05</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Noise-1</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Noise-2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Noise-2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Noise-3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Noise-3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Savage loss appears to be even more robust than \( t \)-logistic loss in a few datasets. However, it is unstable under random initialization of the model parameter. On the second row of each figure, we report the test errors when the model parameter is randomly initialized. We can see that the performance of Savage loss fluctuates in more than half of the datasets. In contrast, \( t \)-logistic loss converges to similar results in all except the longservedio dataset. Therefore, empirically \( t \)-logistic regression appears to be very stable. Logistic regression always converge to the similar result regardless of the initialization because of its convexity.

In order to highlight the main difference between \( t \)-logistic regression and logistic regression, on the third row, we plot the distribution of the forgetting variable \( \xi \) of \( t \)-logistic regression on one of the five folds as \( \ln = 0.10 \). To distinguish the points with noisy labels we plot them in red while the other points in blue. Recall that \( \xi \) denotes the influence of a point. In most of cases, we can observe that the \( \xi \) of the noisy data is smaller than that of the clean data, which indicates that the algorithm is able to effectively identify these noisy points and cap their influence.

7.3.1 Detailed Discussion on Selected Datasets

On the alpha dataset (Figure ??), the performance of the three algorithms are close to each other in the noise-1 model. However, in the noise-3 model, the test error of logistic regression rises from 21.9% of the clean dataset to about 1.1% in the noisy dataset (\( \rho = 0.10 \)). On the other hand, although \( t \)-logistic regression also suffers from unbalanced label noise, the test error rise is only about 0.6%. Similar phenomena are observed on the astro-ph (Figure ??), delta (Figure ??),
epsilon (Figure ??) and gamma (Figure ??) datasets. To understand why \( t \)-logistic regression works better, it is helpful to read its \( \xi \)-distribution on the bottom row. In the noise-3 model of these datasets, the mean of the distribution of the forgetting variable \( \xi \) of the noisy points appears to be much smaller than that of the clean points. Therefore, the influence of those noisy examples is capped. On the other hand, Savage loss also works well in those four datasets when the model parameter is initialized to be all-zero, but its performance sometimes fluctuates with different random initializations, \textit{e.g.} on the delta and gamma datasets.

On the covertype dataset (Figure ??), \( t \)-logistic regression has better test performance with or without label noise in noise-1 and noise-2 model. The reason that \( t \)-logistic regression works better on clean dataset may be because the original dataset is mixed with outliers. The generalization performance of Savage loss is comparable to \( t \)-logistic regression, but it is unstable against random initialization. In the noise-3 model, the performances of all three algorithms becomes worse. For \( t \)-logistic regression, the \( \xi \)-distribution indicates that the influence of half of the noisy examples are not successfully capped.

On the three KDD datasets, kdd99 (Figure ??), kdda (Figure ??), kddb (Figure ??), the number of positive labels is a few times larger than that of the negative labels. Therefore, it appears that all the algorithms perform better in noise-2 and noise-3 model, since the latter contains much fewer noisy examples. \( t \)-logistic regression outperforms logistic regression on the kdd99 and kdda dataset. However, somewhat surprisingly, it performs worse on the kddb dataset although its \( \xi \)-distribution looks similar to that of the kdda dataset.

On the longservedio dataset (Figure ??), all the three algorithms are able to perfectly classify the examples when \( \rho = 0.00 \). In the noise-1 and noise-2 model, \( t \)-logistic regression is apparently more robust against the label noise than logistic regression. In particular, on the \( \xi \)-distribution we observe the 4 distinct spikes. From left to right, the first spike corresponds to the noisy large margin examples, the second spike represents the noisy pullers, the third spike denotes the clean pullers, while the rightmost spike corresponds to the clean large margin examples. On the other hand, logistic regression is unable to discriminate between clean and noisy training samples which leads to its bad performance. In the noise-3 model, more large-margin examples are flipped. Although the overall quantity of noisy examples may be smaller, the impact from these noisy examples is actually strengthened. Not only logistic regression does not perform well in such case, even \( t \)-logistic regression performs much worse. The \( \xi \)-distribution of noise-3 model is apparently different from that of noise-1 and noise-2 model, as there is no spike representing the noisy large margin examples. Furthermore, the flipped large-margin examples clearly create multiple local minima in the empirical risk of \( t \)-logistic regression, as the performance of \( t \)-logistic regression fluctuates with random initialization.

The measowyner dataset (Figure ??) is another dataset where \( t \)-logistic regression demonstrate a clear edge over logistic regression. Here \( t \)-logistic regression outperforms logistic regression in all the three noise models. One can clearly see from the \( \xi \)-distribution that all the red bars lies to the left of the blue bars. The distance between the red bars and the blue bars are even larger in noise-2 and noise-3 model. Similar phenomena are observed on the ocr (Figure ??), reuters-ccat (Figure ??), webspamunigram (Figure ??), worm (Figure ??), zeta (Figure ??) datasets.

The multiclass datasets seem to give a clearer edge to \( t \)-logistic regression. On the letter (Figure ??), sensitacoustic (Figure ??), sensitcombined (Figure ??), sensitseismic (Figure ??) datasets, the test performances of \( t \)-logistic regression are all significantly better than logistic regression even without adding label noise. It is therefore reasonable to conjecture that mis-
labeling is more likely to occur in the multiclass datasets. On the dna (Figure ??), and protein (Figure ??) dataset, t-logistic regression is comparable or slightly better than logistic regression. On the mnist (Figure ??) and usps (Figure ??) dataset, t-logistic regression performs much better when label noises are added.

As is to be expected in such extensive empirical evaluation, there are a few other anomalies. On the aut-avn (Figure ??), dna (Figure ??), real-sim(Figure ??) and webspamtrigram (Figure ??) dataset, logistic regression has the best test accuracy in some of the noise models, although the ξ-distribution indicates that t-logistic regression caps the influence of the noisy examples. On the news20 dataset (Figure ??), the ξ variables of t-logistic regression is almost identical for all examples, which makes its performance close to or not as good as logistic regression. On the beta dataset (Figure ??), the test error of all the algorithms are around 50%.

7.3.2 CPU TIME COMPARISON

One of the drawbacks of t-exponential family is that there is no closed form solution for the log-partition function. The main additional cost of t-logistic regression is the iterative numerical method to compute \( G_t(x; \theta) \) for each example \((x_i, y_i)\). This may impair the efficiency of the algorithm.

In order to compare the time efficiency among the algorithms, we provide the time experiment result of noise-3 model with \( \rho = 0.1 \), containing the total CPU time spent as well as the averaged CPU time spent for every function evaluation in Table ?? and Table ??.

It is not surprising that t-logistic regression takes longer time to train than the logistic regression and the Savage loss in most of the datasets. As the number of the samples is significantly larger than the dimensions e.g. the covertype and kdd99 datasets, the computing cost of \( G_t(x; \theta) \) becomes the primary bottleneck of t-logistic regression and the time efficiency reduces more.

8. Conclusion and Discussions

In this paper, we generalize the logistic regression to the t-logistic regression by using t-exponential family of distributions. The new algorithm has a probabilistic interpretation and is more robust to label noise than logistic regression. We investigate the algorithm in binary classification and multiclass classification. Although the loss function is non-convex, we show that the t-logistic regression is Bayes-risk consistent and empirically stable against random initialization.

In (Ding, 2013), there are two directions that we tried to extend the work of t-logistic regression. The first work uses conditional distributions based on t-exponential family in graphical models, which yields the t-conditional random fields. The second work further generalizes t-logistic regression to obtain a family of loss functions which contains all three types of robustness.

References


t-Logistic Regression


Appendix A. Proof of Theorem 2.3

**Proof** Since the covariance matrix is positive semi-definite, if we show that $\nabla^2 G(\theta) = \text{Var}[\Phi(z)]$, then it automatically implies that $G$ is convex. To show (16) use the regularity condition and expand

$$
\nabla_\theta G(\theta) = \int \Phi(z) \frac{\exp (\Phi(z), \theta)}{\int \exp (\Phi(z), \theta)} d\mathbf{z} = \int \Phi(z)p(\mathbf{z}; \theta) d\mathbf{z} = \mathbb{E} [\Phi(z)].
$$

(49)

Next take the second derivative, use (49), and the definition of variance to write

$$
\nabla^2_\theta G(\theta) = \int \Phi(z) \left[ \Phi(z) - \nabla G(\theta) \right]^\top p(\mathbf{z}; \theta) d\mathbf{z} = \mathbb{E} \left[ \Phi(z) \Phi(z)^\top \right] - \mathbb{E} [\Phi(z)] \mathbb{E} [\Phi(z)]^\top = \text{Var}[\Phi(z)].
$$

\hfill \blacksquare

Appendix B. Proof of Theorem 2.4

**Proof** Because it is unclear how to compute the second derivative of $G_\phi$, we cannot use the same route as in Theorem 2.3 to prove convexity. Therefore the proof relies on more elementary arguments. Recall that $\exp_\phi$ is an increasing and strictly convex function. Choose $\theta_1$ and $\theta_2$ such that $G_\phi(\theta_i) < \infty$ for $i = 1, 2$, and let $\alpha \in (0, 1)$. Set $\theta_\alpha = \alpha \theta_1 + (1 - \alpha) \theta_2$, and observe that

$$
\int \exp_\phi (\langle \Phi(z), \theta_\alpha \rangle - \alpha G_\phi(\theta_1) - (1 - \alpha) G_\phi(\theta_2)) d\mathbf{z} \\
\leq \alpha \int \exp_\phi (\langle \Phi(z), \theta_1 \rangle - G_\phi(\theta_1)) d\mathbf{z} + (1 - \alpha) \int \exp_\phi (\langle \Phi(z), \theta_2 \rangle - G_\phi(\theta_2)) d\mathbf{z} = 1.
$$

On the other hand, we also have

$$
\int \exp_\phi (\langle \Phi(z), \theta_\alpha \rangle - G_\phi(\theta_\alpha)) d\mathbf{z} = 1.
$$

Again, using the fact that $\exp_\phi$ is an increasing function, we can conclude from the above two equations that

$$
G_\phi(\theta_\alpha) \leq \alpha G_\phi(\theta_1) + (1 - \alpha) G_\phi(\theta_2).
$$
This shows that $G_\phi$ is a convex function. We now show (18), using (17) and (14) combined with the fact that $\frac{d}{du} \exp(\phi(u)) = \phi(\exp(\phi(u)))$:

\[
0 \overset{(17)}{=} \int \nabla_\theta p(z; \theta) \, dz = \int \nabla_\theta \exp(\phi(\langle \Phi(z), \theta \rangle - G_\phi(\theta))) \, dz \\
= \int \phi(\exp(\phi(\langle \Phi(z), \theta \rangle - G_\phi(\theta)))) (\Phi(z) - \nabla_\theta G_\phi(\theta)) \, dz \\
\overset{(14)}{=} \left( \int \phi(p(z; \theta)) \, dz \right) \left( \int q(z; \theta) (\Phi(z) - \nabla_\theta G_\phi(\theta)) \, dz \right) \\
= \left( \int \phi(p(z; \theta)) \, dz \right) \left( \mathbb{E}_{q(z; \theta)} [\Phi(z)] - \nabla_\theta G_\phi(\theta) \right).
\]

\[\square\]

Appendix C. Proof of Lemma 4.5

**Proof** For simplicity, let us assume that $\|\theta\| = 1$, then

\[
u = y \langle \Phi(x), \theta \rangle = y\|\Phi(x)\| \cos \psi
\]

\[
\Rightarrow \frac{u}{\cos \psi} = y\|\Phi(x)\|
\]

\[
\Rightarrow \frac{|u|}{|\cos \psi|} = \|\Phi(x)\|
\]

Therefore,

\[
\|\nabla_\theta l(x, y, \theta)\| = \|l'(u) y \Phi(x)\|
\]

\[
= \|l'(u) \frac{u}{\cos \psi}\| = \frac{|I(u)|}{|\cos \psi|}
\]

Since $I(u)$ is bounded, assume that $|I(u)| \leq C$, then,

\[
P(\|\nabla_\theta l(x, y, \theta)\| \to \infty) = P\left( \frac{|I(u)|}{|\cos \psi|} \to \infty \right) \leq P\left( \frac{C}{|\cos \psi|} \to \infty \right) = P(\cos \psi = 0)
\]

Because there is no singleton at $\psi = \frac{\pi}{2}$,

\[
P(\cos \psi = 0) = P(\psi = \frac{\pi}{2}) = 0
\]

\[\square\]
Appendix D. Proof of Lemma 4.6

Proof Using the results in Section D, if $\psi \neq \frac{\pi}{2}$, then as $\|\Phi(x)\| \to \infty$,

$$u = y \langle \Phi(x), \theta \rangle = y\|\Phi(x)\| \cos \psi \to \infty.$$  

Furthermore, since $\lim_{u \to \infty} I(u) = 0$, we have

$$\lim_{\|\Phi(x)\| \to \infty} \|\nabla g l(x, y, \theta)\| = \lim_{u \to \infty} \left| \frac{I(u)}{\cos \psi} \right| = 0.$$  

Because there is no singleton at $\psi = \frac{\pi}{2}$, we conclude

$$P\left( \lim_{\|\Phi(x)\| \to \infty} \|\nabla g l(x, y, \theta)\| = 0 \right) = 1,$$

if $\lim_{u \to \infty} I(u) = 0$.  

Appendix E. Proof of Theorem 4.7

Proof Since $l(u)$ is smooth around $u = 0$, for any given $\epsilon$ there exists $\delta$ such that

$$l'(u) < l'(0) + \epsilon/2 \ , \text{ where } u \in (-\delta, \delta). \quad (50)$$

Define $U = \max \{-u_1, u_2\}$. We construct a set of data points which consist of $x = \{x_1, \ldots, x_{n+1}\}$ with label $y_i = 1$, where $x_1, \ldots, x_n = 1$ and

$$x_{n+1} = \frac{U}{\delta}, \quad n = \frac{l'(0) + \epsilon/2}{l'(0)} x_{n+1}.$$

The gradient of the empirical risk is

$$\nabla R_{emp}(\theta) = \frac{d}{d\theta} \sum_{i=1}^{n} l(\theta x_i) + l(\theta x_{n+1})$$

$$= n \left( l'(\theta) + \frac{n}{x_{n+1}} l'(\theta x_{n+1}) \right)$$

$$= n \left( l'(\theta) - \frac{l'(0)}{l'(0) + \epsilon/2} \left( -\frac{U\theta}{\delta} \right) \right).$$
Now let us investigate the gradient at the following three points $0$, $\delta u_1$, $\delta u_2$.

when $\theta = 0$, $\nabla R_{emp}(0) = n \left( l'(0) - \frac{l'(0)}{l'(0)+\epsilon/2}l'(0) \right) = \frac{nl'(0)e}{2l'(0)+\epsilon} > 0$ (51)

when $\theta = \delta u_1/U$, $\nabla R_{emp}(\delta u_1/U) = n \left( l'(\delta u_1/U) - \frac{l'(0)}{l'(0)+\epsilon/2}l'(u_1) \right)$

$$< n \left( l'(0) + \epsilon/2 - \frac{l'(0)}{l'(0)+\epsilon/2}(l'(0)+\epsilon) \right) = \frac{\epsilon^2}{4(l'(0)+\epsilon)} < 0$$ (52)

when $\theta = \delta u_2/U$, $\nabla R_{emp}(\delta u_2/U) = n \left( l'(\delta u_2/U) - \frac{l'(0)}{l'(0)+\epsilon/2}l'(u_2) \right)$

$$< n \left( l'(0) + \epsilon/2 - \frac{l'(0)}{l'(0)+\epsilon/2}(l'(0)+\epsilon) \right) = \frac{\epsilon^2}{4(l'(0)+\epsilon)} < 0$$ (53)

where the inequalities in (52) and (53) are due to (50) and the definition of $U$.

Therefore, there are at least two $\theta$‘s between $(\delta u_1/U, \delta u_2/U)$ with $\nabla R_{emp}(\theta) = 0$. Since $\nabla R_{emp}(\delta u_1/U) < 0$ and $\nabla R_{emp}(0) > 0$, one local minimum lies in $(\delta u_1/U, 0)$. On the other hand, as $\nabla R_{emp}(\delta u_2/U) < 0$ and the function is lower bounded, the other local minimum lies in $(\delta u_2/U, +\infty)$.

Appendix F. Proof of Theorem 6.1

Proof Since the objective function has a lower bound at 0, we can prove the convergence by showing the algorithm monotonically decreases.

In the $k$-th $\zeta$-step, assuming the current variables are $\theta^{(k-1)}$ and $\zeta^{(k-1)}$, we fix $\theta^{(k-1)}$, denote $\bar{l} = l(\theta^{(k-1)})$, and minimize over $\zeta$. It turns out that:

$$\zeta_{i}^{(k)} = \frac{1}{\bar{l}} \prod_{i=1}^{m} \frac{1}{\bar{l}_i}$$

Therefore,

$$MP(\theta^{(k-1)}, \zeta^{(k)}) = \min_{\zeta} MP(\theta^{(k-1)}, \zeta) = mP(\theta^{(k-1)})^{1/m} \leq MP(\theta^{(k-1)}, \zeta^{(k-1)})$$

The $\theta$-step is to fix $\zeta^{(k)}$ and minimize $\theta$. The result is

$$MP(\theta^{(k)}, \zeta^{(k)}) = \min_{\theta} MP(\theta, \zeta^{(k)}) \leq MP(\theta^{(k-1)}, \zeta^{(k)}) = mP(\theta^{(k-1)})^{1/m}$$

The above two equalities hold if and only if $\zeta^k = \zeta^{k-1}$ and $\theta^k = \theta^{k-1}$, from which the convergence of the algorithm at the $k$-th iteration follows. Therefore, before convergence we have

$$MP(\theta^{(k)}, \zeta^{(k)}) < mP(\theta^{(k-1)})^{1/m} < MP(\theta^{(k-1)}, \zeta^{(k-1)})$$

But since $P(\theta) > 0$, the algorithm must converge at some point.
Next, we show that the converged point $\tilde{\theta}$ is a stationary point of the $P(\theta)$. Assume that $\tilde{\theta}$ and $\tilde{\zeta}$ is the convergence point, then the gradient at the $\theta$-step satisfies:

$$0 = \sum_{i=1}^{m} \zeta_{i} \frac{d l_{i}(\theta)}{d \theta} \bigg|_{\theta=\tilde{\theta}} = \sum_{i=1}^{m} \frac{\prod_{i=1}^{m} l_{i}(\tilde{\theta})^{1/m}}{l_{i}(\tilde{\theta})} \frac{d l_{i}(\theta)}{d \theta} \bigg|_{\theta=\tilde{\theta}}$$

Since $\prod_{i=1}^{m} l_{i}(\tilde{\theta})^{1/m}$ is positive, it implies that,

$$0 = \sum_{i=1}^{m} \frac{\prod_{i=1}^{m} l_{i}(\tilde{\theta})}{l_{i}(\tilde{\theta})} \frac{d l_{i}(\theta)}{d \theta} \bigg|_{\theta=\tilde{\theta}} = d \left( \prod_{i=1}^{m} l_{i}(\theta) \right) \bigg|_{\theta=\tilde{\theta}} = \frac{d P(\theta)}{d \theta} \bigg|_{\theta=\tilde{\theta}}$$

Therefore, $\tilde{\theta}$ is a stationary point of $P(\theta)$.

\[ \blacksquare \]

Appendix G. Verification in Section 3

In this section, we verify that the iterative algorithm for computing $G_{t}$ is going to converge. We only need to verify that $\tilde{a}(k)$ converges to the corresponding $\tilde{a}$ of $\tilde{\alpha}$.

First of all, given $\tilde{a}$, since $t > 1$ and $Z(\tilde{a}) > 1$, it is clear that $0 < \tilde{a}' < \tilde{a}$. On the domain of $0 < \tilde{a}' < \tilde{a}$, it is easy to verify that $Z(\tilde{a}')^{1-t}\tilde{a} - \tilde{a}'$ is a monotonically decreasing function and it crosses at 0 only at $\tilde{a}$. Therefore, when $\tilde{a}(k)$ goes to $\tilde{a}$, we have $\tilde{a}(k+1) < \tilde{a}(k)$ when $\tilde{a}(k) > \tilde{a}$, $\tilde{a}(k+1) > \tilde{a}(k)$.

We then prove that $\tilde{a}(k)$ is a monotonically decreasing sequence. We prove this by mathematical induction. Since $\tilde{a}(0) = \tilde{a}$, $\tilde{a}(1) < \tilde{a} = \tilde{a}(0)$. Next assume that in the $k$-th iteration, $\tilde{a}(k) < \tilde{a}(k-1)$.

Since $Z(\tilde{a}(k)) > Z(\tilde{a}(k-1))$, we have $\tilde{a}(k+1) < \tilde{a}(k)$. Therefore, it follows that $\tilde{a}(k)$ is monotonically decreasing and it is lower bounded by $\tilde{a}$. Furthermore, $\lim_{k \to +\infty} \tilde{a}(k)$ exists.

Finally,

$$\lim_{k \to +\infty} \tilde{a}(k) = \lim_{k \to +\infty} \tilde{a}(k+1) = \lim_{k \to +\infty} Z(\tilde{a}(k))^{1-t}\tilde{a} = Z(\lim_{k \to +\infty} \tilde{a}(k))^{1-t}\tilde{a}, \quad (54)$$

where (54) is because $Z(\tilde{a}')^{1-t}$ is continuous. Therefore, it follows that $\lim_{k \to +\infty} \tilde{a}(k) = \tilde{a}$.

Appendix H. Verification in Section 4.2

In this section, we verify the robust types of the losses in Table 3.

Logistic Regression

$$I_{l}(u) = l'(u)u = -\frac{2}{1 + \exp(2u)}u$$

As $u \to -\infty$, $I_{l}(u)$ goes to infinity. Therefore, logistic regression belongs to Robust Loss 0. Furthermore, one can easily verify that all the convex losses are Robust Loss 0, because $\lim_{u \to -\infty} |l'(u)| \geq |l'(0)| > 0$.  

1040
**T-Logistic Regression** Let us define $p(u) := \exp_t(u - G_t(u))$ and $q(u)$ its escort distribution, then

$$I_t(u) = l'(u)u = -2q(-u)u \cdot p(u)^{t-1}. $$

As $u \to -\infty$, $q(-u) = 1$ and $p(u) = 0$. We have

$$\lim_{u \to -\infty} I_t(u) = \lim_{u \to -\infty} -2u \cdot p(u)^{t-1}$$

$$= \lim_{u \to -\infty} \frac{-2u}{1 + (t-1)(G_t(u) - u)}$$

$$= \lim_{u \to -\infty} \frac{-2}{(t-1)(q(u) - q(-u) - 1)}$$

$$= \lim_{u \to -\infty} \frac{1}{2(t-1)}. \tag{55}$$

where (55) comes by applying the L’Hospital principle.

As $u \to +\infty$, $q(-u) = 0$ and $p(u) = 1$. We have

$$\lim_{u \to +\infty} I_t(u) = \lim_{u \to +\infty} -2u \cdot \frac{p(-u)^t}{p(-u)^t + p(u)^t}$$

$$= \lim_{u \to +\infty} -2up(-u)^{t-1}p(-u)$$

Similar to (56), we have $\lim_{u \to +\infty} -2u \cdot p(-u)^{t-1} = \frac{1}{2(t-1)}$. Furthermore, as $p(-u) = 0$, we conclude that $\lim_{u \to +\infty} I_t(u) = 0$.

Therefore, t-logistic regression belongs to Robust Loss I.

**Savage Loss** The non-convex loss, Savage loss, is widely used in the community of neural network,

$$l(u) = (1 - \sigma(u))^2 = \sigma(-u)^2$$

where $\sigma(u) = \frac{1}{1 + \exp(-u)}$ and $\sigma(u) + \sigma(-u) = 1$.

$$I_t(u) = -2u \cdot \sigma(-u)\sigma'(-u) = 2u \cdot \sigma(-u) \cdot \sigma(-u)\sigma(u)$$

Since, $\lim_{u \to +\infty} \sigma(u) = 0$, and $\lim_{u \to -\infty} \sigma(-u) = 0$, we have $\lim_{u \to \infty} |I_t(u)| = 0$. Therefore, Savage loss belongs to Robust Loss II.

**Appendix I. Verification in Section 5**

In this section, we verify the Bayes-risk consistency property of the multiclass $t$-logistic regression. The Bayes-risk consistency of a multiclass classification loss was first discussed in (Tewari and Bartlett, 2007). Define,

$\hat{a}(x) = (\hat{a}_1, \ldots, \hat{a}_C)$ where, $\hat{a}_c(x) : \mathcal{X} \to \mathbb{R}$ the margin of $x$ in class $c$.

$\eta = (\eta_1, \ldots, \eta_C)$ where, $\eta_c = p(y = c| x)$ the true conditional probability of class $c$.

$l(\hat{a}) = (l_1, \ldots, l_C)$ where, $l_c = l(\hat{a}, c)$
The conditional risk of the multiclass loss $l$ can be written as,

$$C_l(\eta, \hat{a}) = \mathbb{E}_{c|X}[l(\hat{a}, c)] = \sum_{c=1}^{C} \eta_c l_c$$

**Definition 1.1** A Bayes-risk consistent loss function for multiclass classification is the class of loss function $l$, for which given any $\eta$, $\hat{a}^*$ the minimizer of $C_l(\eta, \hat{a})$ satisfies

$$\arg\min_{c} l(\hat{a}^*) \subseteq \arg\max_{c} \eta$$  \hspace{1cm} (57)

For $t$-logistic loss, we have

$$l_c = -\log \exp_t(\hat{a}_c - G_t(\hat{a}))$$

And

$$C_l(\eta, \hat{a}) = \sum_{c=1}^{C} \eta_l l_c$$

$$= -\sum_{c=1}^{C} \eta_c \log \exp_t(\hat{a}_c - G_t(\hat{a}))$$

Minimizing over $\hat{a}$ results in the $\hat{a}^*$ which satisfies $\eta_c = \exp_t(\hat{a}_c^* - G_t(\hat{a}^*))$. Because that $\log$ is a monotonically increasing function,

$$\arg\min_{c} l(\hat{a}^*) = \arg\max_{c} \eta$$  \hspace{1cm} (58)

Therefore, the multiclass $t$-logistic loss is also Bayes-risk consistent.