We extend logistic regression by using t-exponential families which were introduced recently in statistical physics. This gives rise to a regularized risk minimization problem with a non-convex loss function. An efficient block coordinate descent optimization scheme can be derived for estimating the parameters. Because of the nature of the loss function, our algorithm is tolerant to label noise. Furthermore, unlike other algorithms which employ non-convex loss functions, our algorithm is fairly robust to the choice of initial values. We verify both these observations empirically on a number of synthetic and real datasets.

1 Introduction

Many machine learning algorithms minimize a regularized risk [1]:

\[ J(\theta) = \Omega(\theta) + R_{emp}(\theta), \quad \text{where} \quad R_{emp}(\theta) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, \theta). \]  

(1)

Here, \( \Omega \) is a regularizer which penalizes complex \( \theta \); and \( R_{emp} \), the empirical risk, is obtained by averaging the loss \( l \) over the training dataset \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \). In this paper our focus is on binary classification, wherein features of a data point \( x \) are extracted via a feature map \( \phi \) and the label is usually predicted via \( \text{sign}(\langle \phi(x), \theta \rangle) \). If we define the margin of a training example \( (x, y) \) as \( u(x, y, \theta) := y \langle \phi(x), \theta \rangle \), then many popular loss functions for binary classification can be written as functions of the margin. Examples include

\[ l(u) = 0 \text{ if } u > 0 \text{ and } 1 \text{ otherwise .} \]  

(2)

\[ l(u) = \max(0, 1 - u) \]  

(Hinge Loss)  

(3)

\[ l(u) = \exp(-u) \]  

(Exponential Loss)  

(4)

\[ l(u) = \log(1 + \exp(-u)) \]  

(Logistic Loss).  

(5)

The 0–1 loss is non-convex and difficult to handle; it has been shown that it is NP-hard to even approximately minimize the regularized risk with the 0–1 loss [2]. Therefore, other loss functions can be viewed as convex proxies of the 0–1 loss. Hinge loss leads to support vector machines (SVMs), exponential loss is used in Adaboost, and logistic regression uses the logistic loss.

Convexity is a very attractive property because it ensures that the regularized risk minimization problem has a unique global optimum [3]. However, as was recently shown by Long and Servedio [4], learning algorithms based on convex loss functions are not robust to noise\(^2\). Intuitively, the convex loss functions grows at least linearly with slope \( |l'(0)| \) as \( u \in (-\infty, 0) \), which introduces the overwhelming impact from the data with \( u \ll 0 \). There has been some recent and some not-so-recent work on using non-convex loss functions to alleviate the above problem. For instance, a recent manuscript by [5] uses the cdf of the Gaussian distribution to define a non-convex loss.

\(^1\)We slightly abuse notation and use \( l(u) \) to denote \( l(u(x, y, \theta)) \).

\(^2\)Although, the analysis of [4] 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.
In this paper, we continue this line of inquiry and propose 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, a natural extension of exponential family of distributions studied in statistical physics [6–10], we obtain the $t$-logistic regression algorithm. Furthermore, we show that a simple block coordinate descent scheme can be used to solve the resultant regularized risk minimization problem. Analysis of this procedure also intuitively explains why $t$-logistic regression is able to handle label noise.

Our paper is structured as follows: In section 2 we briefly review logistic regression especially in the context of exponential families. In section 3, we review $t$-exponential families, which form the basis for our proposed $t$-logistic regression algorithm introduced in section 4. In section 5 we utilize ideas from convex multiplicative programming to design an optimization strategy. Experiments that compare our new approach to existing algorithms on a number of publicly available datasets are reported in section 6, and the paper concludes with a discussion and outlook in section 7. Some technical details as well as extra experimental results can be found in the supplementary material.

## 2 Logistic Regression

Since we build upon the probabilistic underpinnings of logistic regression, we briefly review some salient concepts. Details can be found in any standard textbook such as [11] or [12]. Assume we are given a labeled dataset $(X, Y) = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ with the $x_i$’s drawn from some domain $\mathcal{X}$ and the labels $y_i \in \{-1, 1\}$. Given a family of conditional distributions parameterized by $\theta$, using Bayes rule, and making a standard iid assumption about the data allows us to write

$$p(\theta \mid X, Y) = p(\theta) \prod_{i=1}^{m} p(y_i \mid x_i; \theta) / p(Y \mid X) \propto p(\theta) \prod_{i=1}^{m} p(y_i \mid x_i; \theta)$$

where $p(Y \mid X)$ is clearly independent of $\theta$. To model $p(y_i \mid x_i; \theta)$, consider the conditional exponential family of distributions

$$p(y \mid x; \theta) = \exp\left(\langle \phi(x, y), \theta \rangle - g(\theta \mid x) \right),$$

with the log-partition function $g(\theta \mid x)$ given by

$$g(\theta \mid x) = \log \left(\exp\left(\langle \phi(x, +1), \theta \rangle \right) + \exp\left(\langle \phi(x, -1), \theta \rangle \right)\right).$$

If we choose the feature map $\phi(x, y) = \frac{y}{2} \phi(x)$, and denote $u = y \langle \phi(x), \theta \rangle$ then it is easy to see that $p(y \mid x; \theta)$ is the logistic function

$$p(y \mid x; \theta) = \frac{\exp(u/2)}{\exp(u/2) + \exp(-u/2)} = \frac{1}{1 + \exp(-u)}.$$  

By assuming a zero mean isotropic Gaussian prior $\mathcal{N}(0, \frac{1}{\sqrt{\lambda}} I)$ for $\theta$, plugging in (9), and taking logarithms, we can rewrite (6) as

$$-\log p(\theta \mid X, Y) = \frac{\lambda}{2} \|\theta\|^2 + \sum_{i=1}^{m} \log \left(1 + \exp\left(-y_i \langle \phi(x_i), \theta \rangle \right)\right) + \text{const.}.$$  

Logistic regression computes a maximum a-posteriori (MAP) estimate for $\theta$ by minimizing (10) as a function of $\theta$. Comparing (1) and (10) it is easy to see that the regularizer employed in logistic regression is $\frac{\lambda}{2} \|\theta\|^2$, while the loss function is the negative log-likelihood $-\log p(y \mid x; \theta)$, which thanks to (9) can be identified with the logistic loss (5).
3 \( t \)-Exponential family of Distributions

In this section we will look at generalizations of the \( \log \) and \( \exp \) functions which were first introduced in statistical physics [6–9]. Some extensions and machine learning applications were presented in [13]. In fact, a more general class of functions was studied in these publications, but for our purposes we will restrict our attention to the so-called \( t \)-exponential and \( t \)-logarithm functions.

The \( t \)-exponential function \( \exp_t \) for \( 0 < t < 2 \) is defined as follows:

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

where \((\cdot)_+ = \max(\cdot, 0)\). Some examples are shown in Figure 2. Clearly, \( \exp_t \) generalizes the usual \( \exp \) function, which is recovered in the limit as \( t \to 1 \). Furthermore, many familiar properties of \( \exp \) are preserved: \( \exp_t \) functions are convex, non-decreasing, non-negative and satisfy \( \exp_t(0) = 1 \).

But \( \exp_t \) does not preserve one very important property of \( \exp \), namely \( \exp_t(a + b) \neq \exp_t(a) \cdot \exp_t(b) \). One can also define the inverse of \( \exp_t \) namely \( \log_t \) as

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

Similarly, \( \log_t(ab) \neq \log_t(a) + \log_t(b) \). From Figure 2, it is clear that \( \exp_t \) decays towards 0 more slowly than the \( \exp \) function for \( 1 < t < 2 \). This important property leads to a family of heavy tailed distributions which we will later exploit.

![Figure 2: Left: \( \exp_t \) and Middle: \( \log_t \) for various values of \( t \) indicated. The right figure depicts the \( t \)-logistic loss functions for different values of \( t \). When \( t = 1 \), we recover the logistic loss.](image)

Analogous to the exponential family of distributions, the \( t \)-exponential family of distributions is defined as [9, 13]:

\[
p(x; \theta) := \exp_t \left( \left( \phi(x), \theta \right) - g_t(\theta) \right). \tag{13}
\]

A prominent member of the \( t \)-exponential family is the Student’s-\( t \) distribution [14]. Just like in the exponential family case, \( g_t \) the log-partition function ensures that \( p(x; \theta) \) is normalized. However, no closed form solution exists for computing \( g_t \) exactly in general. A closely related distribution, which often appears when working with \( t \)-exponential families is the so-called escort distribution [9, 13]:

\[
g_t(x; \theta) := p(x; \theta)^t/Z(\theta) \tag{14}
\]

where \( Z(\theta) = \int p(x; \theta)^t dx \) is the normalizing constant which ensures that the escort distribution integrates to 1.

Although \( g_t(\theta) \) is not the cumulant function of the \( t \)-exponential family, it still preserves convexity. In addition, it is very close to being a moment generating function

\[
\nabla_\theta g_t(\theta) = \mathbb{E}_{q_t(x; \theta)} \left[ \phi(x) \right]. \tag{15}
\]

The proof is provided in the supplementary material. A general version of this result appears as Lemma 3.8 in Sears [13] and a version specialized to the generalized exponential families appears as Proposition 5.2 in [9]. The main difference from \( \nabla_\theta g(\theta) \) of the normal exponential family is that now \( \nabla_\theta g_t(\theta) \) is equal to the expectation of its escort distribution \( g_t(x; \theta) \) instead of \( p(x; \theta) \).
4 Binary Classification with the t-exponential Family

In t-logistic regression we model \( p(y \mid x; \theta) \) via a conditional t-exponential family distribution

\[
p(y \mid x; \theta) = \exp_y \left( \phi(x, y), \theta - g_t(\theta \mid x) \right),
\]

where \( 1 < t < 2 \), and compute the log-partition function \( g_t \) by noting that

\[
\exp_y \left( \phi(x, +1), \theta - g_t(\theta \mid x) \right) + \exp_y \left( \phi(x, -1), \theta - g_t(\theta \mid x) \right) = 1.
\]

Even though no closed form solution exists, one can compute \( g_t \) given \( \theta \) and \( x \) using numerical techniques efficiently.

The Student’s-\( t \) distribution can be regarded as a counterpart of the isotropic Gaussian prior in the t-exponential family [14]. Recall that a one dimensional Student’s-\( t \) distribution is given by

\[
\text{St}(x \mid \mu, \sigma, v) = \frac{\Gamma((v + 1)/2)}{\sqrt{\pi v} \Gamma(v/2) \sigma^{1/2}} \left( 1 + \frac{(x - \mu)^2}{v \sigma^2} \right)^{-(v+1)/2},
\]

where \( \Gamma(\cdot) \) denotes the usual Gamma function and \( v > 1 \) so that the mean is finite. If we select \( t \) satisfying \(- (v + 1)/2 = 1/(1 - t)\) and denote,

\[
\Psi = \frac{\Gamma((v + 1)/2)}{\sqrt{\pi v} \Gamma(v/2) \sigma^{1/2}}^{2/(v+1)},
\]

then by some simple but tedious calculation (included in the supplementary material)

\[
\text{St}(x \mid \mu, \sigma, v) = \exp_y \left( -\hat{\lambda}(x - \mu)^2/2 - \hat{g}_t \right)
\]

where \( \hat{\lambda} = \frac{2\Psi}{(t - 1)\sigma^2} \) and \( \hat{g}_t = \frac{\Psi - 1}{t - 1} \).

Therefore, we work with the Student’s-\( t \) prior in our setting:

\[
p(\theta) = \prod_{j=1}^{d} p(\theta_j) = \prod_{j=1}^{d} \text{St}(\theta_j \mid 0, 2/\lambda, (3 - t)/(t - 1)).
\]

Here, the degree of freedom for Student’s-\( t \) distribution is chosen such that it also belongs to the \( \text{exp}_t \) family, which in turn yields \( v = (3 - t)/(t - 1) \). The Student’s-\( t \) prior is usually preferred to the Gaussian prior when the underlying distribution is heavy-tailed. In practice, it is known to be a robust\(^1\) alternative to the Gaussian distribution [16, 17].

As before, if we let \( \phi(x, y) = \frac{y}{2} \phi(x) \) and plot the negative log-likelihood \(- \log p(y \mid x; \theta)\), then we no longer obtain a convex loss function (see Figure 2). Similarly, \(- \log p(\theta)\) is no longer convex when we use the Student’s-\( t \) prior. This makes optimizing the regularized risk challenging, therefore we employ a different strategy.

Since \( \log_p \) is also a monotonically increasing function, instead of working with \( \log \), we can equivalently work with the \( \log_t \) function (12) and minimize the following objective function:

\[
\hat{J}(\theta) = - \log_t p(\theta) \prod_{i=1}^{m} p(y_i \mid x_i; \theta) / p(Y \mid X)
\]

\[
= \frac{1}{t - 1} \left( p(\theta) \prod_{i=1}^{m} p(y_i \mid x_i; \theta) / p(Y \mid X) \right)^{1-t} + \frac{1}{1-t},
\]

where \( p(Y \mid X) \) is independent of \( \theta \). Using (13), (18), and (11), we can further write

\[
\hat{J}(\theta) \propto \prod_{j=1}^{d} \left( 1 + (1 - t)\left(-\hat{\lambda} \theta_j^2/2 - \hat{g}_t \right) \right) \prod_{i=1}^{m} \left( 1 + (1 - t)\left( \frac{y_i}{2} \phi(x_i, \theta) - g_t(\theta \mid x_i) \right) \right) + \text{const.}
\]

\[
= \prod_{j=1}^{d} r_j(\theta) \prod_{i=1}^{m} l_i(\theta) + \text{const.}
\]

\(^1\)There is no unique definition of robustness. For example, one of the definitions is through the outlier-proneness [15]: \( p(\theta \mid X, Y, x_{n+1}, y_{n+1}) \rightarrow p(\theta \mid X, Y) \) as \( x_{n+1} \rightarrow \infty \).
Since \( t > 1 \), it is easy to see that \( r_{j}(\theta) > 0 \) is a convex function of \( \theta \). On the other hand, since \( g_{t} \) is convex and \( t > 1 \) it follows that \( l_{1}(\theta) > 0 \) is also a convex function of \( \theta \). In summary, \( J(\theta) \) is a product of positive convex functions. In the next section we will present an efficient optimization strategy for dealing with such problems.

## 5 Convex Multiplicative Programming

In convex multiplicative programming [18] we are interested in the following optimization problem:

\[
\min_{\theta} \quad P(\theta) \triangleq \prod_{n=1}^{N} z_{n}(\theta) \quad \text{s.t.} \quad \theta \in \mathbb{R}^{d},
\]

(23)

where \( z_{n}(\theta) \) are positive convex functions. Clearly, \( (\ref{eq:23}) \) can be identified with \( (\ref{eq:22}) \) by setting \( N = d + m \) and identifying \( z_{n}(\theta) = r_{n}(\theta) \) for \( n = 1, \ldots, d \) and \( z_{n+d}(\theta) = l_{n}(\theta) \) for \( n = 1, \ldots, m \).

The optimal solutions to the problem \( (\ref{eq:23}) \) can be obtained by solving the following parametric problem (see Theorem 2.1 of Kuno et al. [18]):

\[
\min_{\xi} \min_{\theta} \quad \text{MP}(\theta, \xi) \triangleq \sum_{n=1}^{N} \xi_{n} z_{n}(\theta) \quad \text{s.t.} \quad \theta \in \mathbb{R}^{d}, \quad \xi > 0, \quad \prod_{n=1}^{N} \xi_{n} \geq 1.
\]

(24)

The optimization problem in \( (\ref{eq:24}) \) is very reminiscent of logistic regression. In logistic regression, \( l_{n}(\theta) = -\langle \frac{1}{d} \phi(x_{n}), \theta \rangle + g(\theta | x_{n}) \), while here \( l_{n}(\theta) = 1 + (1 - t) \langle \frac{1}{d} \phi(x_{n}), \theta \rangle - g_{t}(\theta | x_{n}) \).

The key difference is that in \( t \)-logistic regression each data point \( x_{n} \) has a weight (or influence) \( \xi_{n} \) associated with it.

Exact algorithms have been proposed for solving \( (\ref{eq:24}) \) (for instance, [18]). However, the computational cost of these algorithms grows exponentially with respect to \( N \) which makes them impractical for our purposes. Instead, we apply a block coordinate descent based method. The main idea is to minimize \( (\ref{eq:24}) \) with respect to \( \theta \) and \( \xi \) separately.

**\( \xi \)-Step:** Assume that \( \theta \) is fixed, and denote \( \tilde{z}_{n} = z_{n}(\theta) \) to rewrite \( (\ref{eq:24}) \) as:

\[
\min_{\xi} \quad \sum_{n=1}^{N} \xi_{n} \tilde{z}_{n} \quad \text{s.t.} \quad \xi > 0, \quad \prod_{n=1}^{N} \xi_{n} \geq 1.
\]

(25)

Since the objective function is linear in \( \xi \) and the feasible region is a convex set, \( (\ref{eq:25}) \) is a convex optimization problem. By introducing a non-negative Lagrange multiplier \( \gamma \geq 0 \), the partial Lagrangian and its gradient with respect to \( \xi_{n} \) can be written as

\[
L(\xi, \gamma) = \sum_{n=1}^{N} \xi_{n} \tilde{z}_{n} + \gamma \cdot \left( 1 - \prod_{n=1}^{N} \xi_{n} \right)
\]

(26)

\[
\frac{\partial}{\partial \xi_{n'}} L(\xi, \gamma) = \tilde{z}_{n'} - \gamma \prod_{n \neq n'} \xi_{n}.
\]

(27)

Setting the gradient to 0 obtains \( \gamma = \frac{\tilde{z}_{n'}}{\prod_{n \neq n'} \xi_{n}} \). Since \( \tilde{z}_{n'} > 0 \), it follows that \( \gamma \) cannot be 0. By the K.K.T. conditions [3], we can conclude that \( \prod_{n=1}^{N} \xi_{n} = 1 \). This in turn implies that \( \gamma = \tilde{z}_{n} \) or

\[
(\xi_{1}, \ldots, \xi_{N}) = (\gamma / \tilde{z}_{1}, \ldots, \gamma / \tilde{z}_{N}), \quad \text{with} \quad \gamma = \prod_{n=1}^{N} \tilde{z}_{n}^{-1}.
\]

(28)

Recall that \( \xi_{n} \) in \( (\ref{eq:24}) \) is the weight (or influence) of each term \( z_{n}(\theta) \). The above analysis shows that \( \gamma = \tilde{z}_{n}(\theta) \xi_{n} \) remains constant for all \( n \). If \( \tilde{z}_{n}(\theta) \) becomes very large then its influence \( \xi_{n} \) is reduced. Therefore, points with very large loss have their influence capped and this makes the algorithm robust to outliers.

**\( \theta \)-Step:** In this step we fix \( \xi > 0 \) and solve for the optimal \( \theta \). This step is essentially the same as logistic regression, except that each component has a weight \( \xi \) here.

\[
\min_{\theta} \quad \sum_{n=1}^{N} \xi_{n} z_{n}(\theta) \quad \text{s.t.} \quad \theta \in \mathbb{R}^{d}.
\]

(29)
In Figure 3, we plot the test error with and without label noise. In the latter case, the test error of experiments were performed on a quad-core machine with Dual 2.5 Ghz processor and 32 Gb RAM. All code is written in Matlab, and for the linear SVM we use the Matlab interface of LibSVM [22].

The objective function is monotonically decreasing and is guaranteed to converge to a stable point of $P(\theta)$. We include the proof in the supplementary material.

6 Experimental Evaluation

Our experimental evaluation is designed to answer four natural questions: 1) How does the generalization capability (measured in terms of test error) of $t$-logistic regression compare with existing algorithms such as logistic regression and support vector machines (SVMs) both in the presence and absence of label noise? 2) Do the $\xi$ variables we introduced in the previous section have a natural interpretation? 3) How much overhead does $t$-logistic regression incur as compared to logistic regression? 4) How sensitive is the algorithm to initialization? The last question is particularly important given that the algorithm is minimizing a non-convex loss.

To answer the above questions empirically we use six datasets, two of which are synthetic. The Long-Servedio dataset is an artificially constructed dataset to show that algorithms which minimize a differentiable convex loss are not tolerant to label noise Long and Servedio [4]. The examples have 21 dimensions and play one of three possible roles: large margin examples (25%, $x_{1,2,...,21} = y$); pullers (25%, $x_1,...,11 = y$, $x_{12,...,21} = -y$); 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 $-y$). The Mease-Wyner is another synthetic dataset to test the effect of label noise. The input $x$ is a 20-dimensional vector where each coordinate is uniformly distributed on $[0, 1]$. The label $y$ is $+1$ if $\sum_{j=1}^{5} x_j \geq 2.5$ and $-1$ otherwise [19]. In addition, we also test on Mushroom, USPS-N (9 vs. others), Adult, and Web datasets, which are often used to evaluate machine learning algorithms (see Table 1 in supplementary material for details).

For simplicity, we use the identity feature map $\phi(x) = x$ in all our experiments, and set $t \in \{1.3, 1.6, 1.9\}$ for $t$-logistic regression. Our comparators are logistic regression, linear SVMs\(^4\), and an algorithm (the probit) which employs the probit loss, $\mathcal{L}(u) = 1 - erf(2u)$, used in Brown-Boost/RobustBoost [5]. We use the L-BFGS algorithm [21] for the $\theta$-step in $t$-logistic regression. L-BFGS is also used to train logistic regression and the probit loss based algorithms. Label noise is added by randomly choosing 10% of the labels in the training set and flipping them; each dataset is tested with and without label noise. We randomly select and hold out 30% of each dataset as a validation set and use the rest of the 70% for 10-fold cross validation. The optimal parameters namely $\lambda$ for $t$-logistic and logistic regression and $C$ for SVMs is chosen by performing a grid search over the parameter space $\{2^{-7}, -6,..., 7\}$ and observing the prediction accuracy over the validation set. The convergence criterion is to stop when the change in the objective function value is less than $10^{-4}$.

All code is written in Matlab, and for the linear SVM we use the Matlab interface of LibSVM [22]. Experiments were performed on a quad-core machine with Dual 2.5 Ghz processor and 32 Gb RAM.

\(^4\)We also experimented with RampSVM [20], however, the results are worse than the other algorithms. We therefore report these results in the supplementary material.
Figure 3: The test error rate of various algorithms on six datasets (left to right, top: Long-Servedio, Mease-Wyner, Mushroom; bottom: USPS-N, Adult, Web) with and without 10% label noise. All algorithms are initialized with $\theta = 0$. The blue (light) bar denotes a clean dataset while the magenta (dark) bar are the results with label noise added. Also see Table 3 in the supplementary material.

Long-Servedio and Mushroom datasets), with a slight edge on some datasets such as Mease-Wyner. When label noise is added, $t$-logistic regression (especially with $t = 1.9$) shows significantly better performance than all the other algorithms on all datasets except the USPS-N, where it is marginally outperformed by the probit.

To obtain Figure 4 we used the noisy version of the datasets, chose one of the 10 folds used in the previous experiment, and plotted the distribution of the $1/\xi$ obtained after training with $t = 1.9$. To distinguish the points with noisy labels we plot them in cyan while the other points are plotted in red. Analogous plots for other values of $t$ can be found in the supplementary material. Recall that $\xi$ denotes the influence of a point. One can clearly observe that the $\xi$ of the noisy data is much smaller than that of the clean data, which indicates that the algorithm is able to effectively identify these points and cap their influence. In particular, on the Long-Servedio dataset 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. Clearly, the noisy large margin examples and the noisy pullers are assigned a low value of $\xi$ thus capping their influence and leading to the perfect classification of the test set. On the other hand, logistic regression is unable to discriminate between clean and noisy training samples which leads to bad performance on noisy datasets.

Detailed timing experiments can be found in Table 4 in the supplementary material. In a nutshell, $t$-logistic regression takes longer to train than either logistic regression or the probit. The reasons are not difficult to see. First, there is no closed form expression for $g_t(\theta \mid x)$. We therefore resort to pre-computing it at some fixed locations and using a spline method to interpolate values at other locations. Second, since the objective function is not convex several iterations of the $\xi$ and $\theta$ steps might be needed. Surprisingly, the L-BFGS algorithm, which is not designed to optimize non-convex functions, is able to minimize (22) directly in many cases. When it does converge, it is often faster than the convex multiplicative programming algorithm. However, on some cases (as expected) it fails to find a direction of descent and exits. A common remedy for this is the bundle L-BFGS with a trust-region approach. [21]

Given that the $t$-logistic objective function is non-convex, one naturally worries about how different initial values affect the quality of the final solution. To answer this question, we initialized the algorithm with 50 different randomly chosen $\theta \in [-0.5, 0.5]^d$, and report test performances of the various solutions obtained in Figure 5. Just like logistic regression which uses a convex loss and hence converges to the same solution independent of the initialization, the solution obtained

---

3We provide the significance test results in Table 2 of supplementary material.
Figure 4: The distribution of $\xi$ obtained after training $t$-logistic regression with $t = 1.9$ on datasets with 10% label noise. Left to right, top: Long-Servedio, Mease-Wyner, Mushroom; bottom: USPS-N, Adult, Web. The red (dark) bars (resp. cyan (light) bars) indicate the frequency of $\xi$ assigned to points without (resp. with) label noise.

by $t$-logistic regression seems fairly independent of the initial value of $\theta$. On the other hand, the performance of the probit fluctuates widely with different initial values of $\theta$.

Figure 5: The Error rate by different initialization. Left to right, top: Long-Servedio, Mease-Wyner, Mushroom; bottom: USPS-N, Adult, Web.

7 Discussion and Outlook

In this paper, we generalize logistic regression to $t$-logistic regression by using the $t$-exponential family. The new algorithm has a probabilistic interpretation and is more robust to label noise. Even though the resulting objective function is non-convex, empirically it appears to be insensitive to initialization. There are a number of avenues for future work. On Long-Servedio experiment, if the label noise is increased significantly beyond 10%, the performance of $t$-logistic regression may degrade (see Fig. 6 in supplementary materials). Understanding and explaining this issue theoretically and empirically remains an open problem. It will be interesting to investigate if $t$-logistic regression can be married with graphical models to yield $t$-conditional random fields. We will also focus on better numerical techniques to accelerate the $\theta$-step, especially a faster way to compute $g_t$. 
References


A  Student’s-t Distribution

Recall that a $k$-dimensional Student’s-t distribution $St(x|\mu, \Sigma, v)$ with $1 < v < +\infty$ degrees of freedom has the following probability density function:

$$St(x|\mu, \Sigma, v) = \frac{\Gamma((v + k)/2)}{(\pi v)^{k/2} \Gamma(v/2)|\Sigma|^{1/2}} \left(1 + (x - \mu)^\top (\Sigma)^{-1} (x - \mu)\right)^{-\frac{v+k}{2}}. \quad (31)$$

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

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

to rewrite (31) as

$$St(x|\mu, \Sigma, v) = \left(\Psi + \Psi \cdot (x - \mu)^\top (\Sigma)^{-1} (x - \mu)\right)^{1/(1-t)}. \quad (32)$$

Next we set $\phi(x) = [x; xx^\top]$, $\theta = [\theta_1, \theta_2]$, where $K = (\Sigma)^{-1}$ and $\theta_1 = -2\Psi K\mu/(1-t)$ while $\theta_2 = \Psi K/(1-t)$. Then we define

$$\langle \phi(x), \theta \rangle = \left(\frac{\Psi}{1-t}\right) (x^\top K x - 2\mu^\top K x)$$

and

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

to rewrite (32) as

$$St(x|\mu, \Sigma, v) = (1 + (1-t)\left(\langle \phi(x), \theta \rangle - g_t(\theta)\right))^{1/(1-t)}.$$}

Comparing with (11) clearly shows that

$$St(x|\mu, \Sigma, v) = p_t(x; \theta) = \exp_t (\langle \phi(x), \theta \rangle - g_t(\theta)).$$

Furthermore, using this fact and some simple algebra yields the escort distribution of Student’s-t distribution:

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

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.

B  Properties of $g_t(\theta)$

Although $g_t(\theta)$ is not the cumulant function of the $t$-exponential family, it still preserves convexity. As the following theorem asserts, its first derivative can still be written as an expectation of $\phi(x)$ but now with respect to the escort distribution. Note that the theorem and proof here is only a special case of a more general one that appears in Sears [13] and Naudts [9].

Theorem 1 The function $g_t(\theta)$ is convex. Moreover, if the following regularity condition

$$\int \nabla_\theta p(x; \theta) dx = \nabla_\theta \int p(x; \theta) dx \quad (33)$$

holds, then

$$\nabla_\theta g_t(\theta) = E_{q_t(x; \theta)} [\phi(x)], \quad (34)$$

where $q_t(x; \theta)$ is the escort distribution (14).
Proof To prove convexity, we rely on the elementary arguments. Recall that $\exp_t$ is an increasing and strictly convex function. Choose $\theta_1$ and $\theta_2$ such that $g_t(\theta_i) < \infty$ for $i = 1, 2$, and let $\alpha \in (0, 1)$. Set $\theta_\alpha = \alpha \theta_1 + (1 - \alpha) \theta_2$, and observe that

$$\int \exp_t((\phi(x), \theta_\alpha) - \alpha g_t(\theta_1) - (1 - \alpha) g_t(\theta_2)) \, dx$$

$$< \alpha \int \exp_t((\phi(x), \theta_1) - g_t(\theta_1)) \, dx + (1 - \alpha) \int \exp_t((\phi(x), \theta_2) - g_t(\theta_2)) \, dx = 1.$$ 

On the other hand, we also have

$$\int \exp_t((\phi(x), \theta_\alpha) - g_t(\theta_\alpha)) \, dx = 1.$$ 

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

$$g_t(\theta_\alpha) < \alpha g_t(\theta_1) + (1 - \alpha) g_t(\theta_2).$$

This shows that $g_t$ is a strictly convex function. To show (34) use (33) and observe that

$$(\int q_t(x; \theta) dx = 1)$$

Combining with the fact that $\frac{d}{dt} \exp_t(x) = \exp_t(x)$, use (14) and the chain rule to write

$$\int \nabla_\theta p(x; \theta) dx = \int \nabla_\theta \exp_t((\phi(x), \theta) - g_t(\theta)) \, dx$$

$$= \int \exp_t((\phi(x), \theta) - g_t(\theta)) (\phi(x) - \nabla_\theta g_t(\theta)) dx$$

$$= \int q_t(x; \theta)(\phi(x) - \nabla_\theta g_t(\theta)) dx = 0.$$ 

Rearranging terms and using $\int q_t(x; \theta) dx = 1$ directly yields (34). \qedsymbol

C Convergence of Convex Multiplicative Programming

In the Convex Multiplicative Programming, we convert the problem:

$$\arg\min_\theta P(\theta) \triangleq \prod_{n=1}^{N} z_n(\theta)$$

into the problem:

$$\arg\min_{\theta, \xi} \text{MP}(\theta, \xi) \triangleq \sum_{n=1}^{N} \xi_n z_n(\theta) \quad \text{s.t.} \quad \prod_{n=1}^{N} \xi_n = 1 \text{ and } \xi > 0$$

by introducing the latent variable $\xi$. In the $k$-th step, assuming the current variables are $\theta^{(k-1)}$ and $\xi^{(k-1)}$, we fix $\theta^{(k-1)}$, denote $\tilde{z} = z(\theta^{(k-1)})$, and minimize over $\xi$. It turns out that:

$$\xi_n^{(k)} = \frac{1}{\tilde{z}_n} \prod_{n=1}^{N} \tilde{z}_n^{\frac{1}{N}}$$

Therefore,

$$\text{MP}(\theta^{(k-1)}, \xi^{(k)}) = \min_{\xi} \text{MP}(\theta^{(k-1)}, \xi) = \text{NP}(\theta^{(k-1)})^{1/N} \leq \text{MP}(\theta^{(k-1)}, \xi^{(k-1)})$$
The $\theta$-step is to fix $\xi^{(k)}$ and minimize $\theta$, the result is

$$
\text{MP}(\theta^{(k)}, \xi^{(k)}) = \min_{\theta} \text{MP}(\theta, \xi^{(k)}) \leq \text{MP}(\theta^{(k-1)}, \xi^{(k)}) = \text{NP}(\theta^{(k-1)})^{1/N}
$$

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

$$
\text{MP}(\theta^{(k)}, \xi^{(k)}) < \text{NP}(\theta^{(k-1)})^{1/N} < \text{MP}(\theta^{(k-1)}, \xi^{(k-1)}).
$$

But since $P(\theta) > 0$, the algorithm must converge at some point.

Next, we want to show that $\tilde{\theta}$ after convergence is a stable point of the $P(\theta)$. Assume that $\tilde{\theta}$ and $\tilde{\xi}$ is the convergence point, then the $\theta$-step is:

$$
0 = \sum_{n=1}^{N} \prod_{n=1}^{N} \frac{1}{z_n(\theta)} \frac{dz_n(\theta)|_{\theta=\tilde{\theta}}}{d\theta} = \left( \prod_{n=1}^{N} z_n(\theta) \right) \left( \sum_{n=1}^{N} \frac{1}{z_n(\theta)} \frac{dz_n(\theta)|_{\theta=\tilde{\theta}}}{d\theta} \right)
$$

which implies that,

$$
0 = \sum_{n=1}^{N} \frac{1}{z_n(\theta)|_{\theta=\tilde{\theta}}} \frac{dz_n(\theta)|_{\theta=\tilde{\theta}}}{d\theta} = \sum_{n=1}^{N} \prod_{n=1}^{N} z_n(\tilde{\theta}) \frac{dz_n(\theta)|_{\theta=\tilde{\theta}}}{d\theta} = \frac{dP(\theta)|_{\theta=\tilde{\theta}}}{d\theta}
$$

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

## D Gradient Based method

It is also possible to directly use the gradient based method such as L-BFGS to solve (23). To do this, it is convenient to take log of (23).

$$
\log P(\theta) = \sum_{n=1}^{N} \log z_n(\theta) = \sum_{j=1}^{d} \log r_j(\theta) + \sum_{i=1}^{m} \log l_i(\theta) = \sum_{j=1}^{d} \log \left(1 + (1 - t)(-\tilde{\lambda}\theta_j^2/2 - \tilde{g}_j)\right) + \sum_{i=1}^{m} \log \left(1 + (1 - t)(\tilde{\phi}(x_i, \theta) - g_i | x_i)\right)
$$

(35)

Take the derivative,

$$
\nabla_{\theta} \log z_n(\theta) = \nabla_{\theta} \log r_n(\theta) = \left( t - 1 \right) \cdot \tilde{\lambda} \cdot e_n
$$

(37)

for $n = 1, \ldots, d$ \quad $\nabla_{\theta} \log z_{n+d}(\theta) = \nabla_{\theta} \log l_n(\theta)$

$$
\left( 1 - t \right) \cdot \left[ \frac{\tilde{y}_n}{2} \cdot \phi(x_n) - \mathbb{E}_{q_n}(y_n | x_n, \theta) \left[ \frac{\tilde{y}_n}{2} \cdot \phi(x_n) \right] \right]
$$

(38)

where $e_n$ denotes the $d$ dimensional vector with one at the $n$-th coordinate and zeros elsewhere ($n$-th unit vector). There is an obvious relation of (37) (38), and the previous routine, given that $\xi_n = 1/z_n(\theta)$ here and $\xi_n \propto 1/\tilde{z}_n$ in (29).

We report the performance of the $t$-logistic regression by directly using L-BFGS as the optimizer in Table 5. The algorithms use the same parameters as in Table 1. Since L-BFGS is not designed to optimize non-convex functions, it may fail sometimes, in which case we randomly restart with a different initialization.

## E Higher Label Noise

Our algorithm appears to be more robust than logistic regression (especially when $t = 1.9$) against the label noise (10%). A natural question to ask then is how well it performs when the label noise is larger than 10%. In Figure 6, we compare the $t$-logistic regression with logistic regression and the probit in the cases when 20% and 30% label noise is added. We also report the test error in Table 6.
Figure 6: The test performance with the change of the label noise (left to right, top: Long-Servedio, Mease-Wyner, Mushroom; bottom: USPS-N, Adult, Web). The magenta dash line with upper-triangles is the logistic regression; the green line with circles is $t = 1.3$; the cyan line with squares is $t = 1.6$; the red line with diamonds is $t = 1.9$; and the blue dash line with lower-triangles is the probit.

F Significance Test

We performed the paired $t$-test of the test error rates for each dataset obtained by the $t$-logistic regression with $t = 1.9$ and by the other algorithms. To do this, we take the difference of the error rate for each split in each dataset by any two algorithms. The hypothesis is that the difference of the two algorithms for each split is drawn from a zero-mean normal distribution with unknown variance in the same dataset. We report results in Table 2.

G Selected Results from RampSVM

Unfortunately, we do not obtain good results by using RampSVM [20] on our datasets. We used the UniverSVM package [23], and performed a grid search of the parameters $C$ and $s$. It appears that the optimal parameter $C$ is consistent with that used by the Linear SVM, since the UniverSVM uses the solution of Linear SVM to initialize. We report the test performance for $s = 0, -0.1, -1, -10, -100$ in Table 7. Usually the results are significantly worse than the other algorithms, except for the Long-Servedio dataset where RampSVM performs as well as other non-convex losses with label noise. We therefore do not report results the RampSVM in the main body of the paper.
Table 1: Datasets used in our experiments. (\(\lambda_t\) denotes \(\lambda\) for the logistic regression. \(\lambda_{1.3}, \lambda_{1.6}, \lambda_{1.9}\) are the \(\lambda\) values used for \(t\)-logistic regression with \(t = 1.3, 1.6, \) and \(1.9\) respectively. \(\lambda_p\) is \(\lambda\) for the probit algorithm. \(C\) is the parameter \(C\) for the C-SVC which is equivalent to \(1/\lambda_p\).)

<table>
<thead>
<tr>
<th>Name</th>
<th>Noise</th>
<th>Dimensions</th>
<th>Num. of examples</th>
<th>(\lambda_t)</th>
<th>(\lambda_{1.3})</th>
<th>(\lambda_{1.6})</th>
<th>(\lambda_{1.9})</th>
<th>(\lambda_p)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-Servedio</td>
<td>0%</td>
<td>21</td>
<td>2000</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>0%</td>
<td>20</td>
<td>2000</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Mushroom</td>
<td>0%</td>
<td>112</td>
<td>8124</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>USPS-N</td>
<td>0%</td>
<td>256</td>
<td>11000</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{3})</td>
<td></td>
</tr>
<tr>
<td>Adult</td>
<td>0%</td>
<td>123</td>
<td>48842</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>0%</td>
<td>300</td>
<td>64700</td>
<td>(2^{-5})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td></td>
</tr>
<tr>
<td>Long-Servedio</td>
<td>10%</td>
<td>21</td>
<td>2000</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-1})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>10%</td>
<td>20</td>
<td>2000</td>
<td>(2^{-2})</td>
<td>(2^{-2})</td>
<td>(2^{-2})</td>
<td>(2^{-2})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Mushroom</td>
<td>10%</td>
<td>112</td>
<td>8124</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{1})</td>
<td>(2^{3})</td>
<td></td>
</tr>
<tr>
<td>USPS-N</td>
<td>10%</td>
<td>256</td>
<td>11000</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{2})</td>
<td>(2^{-3})</td>
<td></td>
</tr>
<tr>
<td>Adult</td>
<td>10%</td>
<td>123</td>
<td>48842</td>
<td>(2^{4})</td>
<td>(2^{4})</td>
<td>(2^{4})</td>
<td>(2^{4})</td>
<td>(2^{-1})</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>10%</td>
<td>300</td>
<td>64700</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td>(2^{-7})</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Significance Test of the test error rates by the \(t\)-logistic regression with \(t = 1.3\) and the other algorithms. The significance factor \(\alpha\) is set as 0.05. 'Y' means that the difference is significant. 'N' means the difference is not significant.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Noise</th>
<th>Logistic</th>
<th>(t=1.3)</th>
<th>(t=1.6)</th>
<th>Probit</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-Servedio</td>
<td>0%</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>0%</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0%</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>USPS-N</td>
<td>0%</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Adult</td>
<td>0%</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Web</td>
<td>0%</td>
<td></td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Long-Servedio</td>
<td>10%</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>10%</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Mushroom</td>
<td>10%</td>
<td></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>USPS-N</td>
<td>10%</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Adult</td>
<td>10%</td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Web</td>
<td>10%</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 7: The \(\xi\) distribution with 10% label noise added. \(t = 1.3\). Left to right, top: Long-Servedio, Mease-Wyner, Mushroom; bottom: USPS-N, Adult, Web. The red bars (resp. cyan bars) indicate the \(\xi\) assigned to points without (resp. with) label noise.
Table 3: Test Error Rate in % for various algorithms. The noisy version is obtained by flipping the labels of randomly chosen fixed fraction of the training data.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Noise</th>
<th>Logistic</th>
<th>Probit</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-Servedio</td>
<td>0%</td>
<td>0.02 ± 0.00</td>
<td>0.17 ± 0.12</td>
<td>0.18 ± 0.03</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>0%</td>
<td>0.02 ± 0.01</td>
<td>1.12 ± 0.30</td>
<td>1.70 ± 0.46</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0%</td>
<td>0.33 ± 0.03</td>
<td>0.77 ± 0.10</td>
<td>1.89 ± 0.28</td>
</tr>
<tr>
<td>USPS-N</td>
<td>0%</td>
<td>0.47 ± 0.11</td>
<td>7.09 ± 2.20</td>
<td>12.65 ± 3.05</td>
</tr>
<tr>
<td>Adult</td>
<td>0%</td>
<td>1.03 ± 0.06</td>
<td>11.11 ± 4.70</td>
<td>27.73 ± 13.10</td>
</tr>
<tr>
<td>Web</td>
<td>0%</td>
<td>4.02 ± 0.36</td>
<td>41.52 ± 9.01</td>
<td>31.40 ± 11.14</td>
</tr>
</tbody>
</table>

Table 4: CPU Time (in seconds)
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Noise</th>
<th>Test Error (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>t=1.3</td>
<td>t=1.6</td>
</tr>
<tr>
<td>Long-Servedio</td>
<td>0%</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>0%</td>
<td>0.36 ± 0.51</td>
<td>0.29 ± 0.50</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0%</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>USPS-N</td>
<td>0%</td>
<td>2.45 ± 0.59</td>
<td>2.48 ± 0.62</td>
</tr>
<tr>
<td>Adult</td>
<td>0%</td>
<td>15.27 ± 0.62</td>
<td>15.30 ± 0.60</td>
</tr>
<tr>
<td>Web</td>
<td>0%</td>
<td>1.31 ± 0.15</td>
<td>1.40 ± 0.16</td>
</tr>
<tr>
<td>Long-Servedio</td>
<td>10%</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>10%</td>
<td>2.00 ± 1.00</td>
<td>1.00 ± 0.90</td>
</tr>
<tr>
<td>Mushroom</td>
<td>10%</td>
<td>0.12 ± 0.17</td>
<td>0.11 ± 0.12</td>
</tr>
<tr>
<td>USPS-N</td>
<td>10%</td>
<td>3.22 ± 0.93</td>
<td>2.92 ± 0.85</td>
</tr>
<tr>
<td>Adult</td>
<td>10%</td>
<td>15.36 ± 0.53</td>
<td>15.34 ± 0.49</td>
</tr>
<tr>
<td>Web</td>
<td>10%</td>
<td>1.50 ± 0.18</td>
<td>1.47 ± 0.17</td>
</tr>
</tbody>
</table>

Table 5: Results of $t$-logistic regression by using L-BFGS directly.
Table 6: Test Error Rate in % for various algorithms with higher label noise.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Noise</th>
<th>Logistic</th>
<th>t=1.3</th>
<th>t=1.6</th>
<th>t=1.9</th>
<th>Probit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-Servedio</td>
<td>20%</td>
<td>26.14 ± 5.39</td>
<td>25.43 ± 5.25</td>
<td>23.64 ± 5.49</td>
<td>9.93 ± 12.97</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>20%</td>
<td>7.86 ± 2.36</td>
<td>5.86 ± 1.46</td>
<td>4.07 ± 1.91</td>
<td>3.07 ± 1.47</td>
<td>3.00 ± 1.38</td>
</tr>
<tr>
<td>Mushroom</td>
<td>20%</td>
<td>0.21 ± 0.27</td>
<td>0.19 ± 0.24</td>
<td>0.19 ± 0.24</td>
<td>0.19 ± 0.24</td>
<td>0.95 ± 0.45</td>
</tr>
<tr>
<td>USPS-N</td>
<td>20%</td>
<td>4.82 ± 0.84</td>
<td>4.73 ± 0.82</td>
<td>4.21 ± 0.82</td>
<td>3.77 ± 0.68</td>
<td>3.23 ± 0.79</td>
</tr>
<tr>
<td>Adult</td>
<td>20%</td>
<td>15.59 ± 0.49</td>
<td>15.56 ± 0.50</td>
<td>15.51 ± 0.50</td>
<td>15.47 ± 0.45</td>
<td>15.65 ± 0.42</td>
</tr>
<tr>
<td>Web</td>
<td>20%</td>
<td>1.77 ± 0.21</td>
<td>1.69 ± 0.21</td>
<td>1.64 ± 0.21</td>
<td>1.59 ± 0.19</td>
<td>3.02 ± 0.23</td>
</tr>
<tr>
<td>Long-Servedio</td>
<td>30%</td>
<td>26.71 ± 4.34</td>
<td>26.57 ± 4.43</td>
<td>26.43 ± 4.48</td>
<td>26.07 ± 4.55</td>
<td>2.86 ± 2.54</td>
</tr>
<tr>
<td>Mease-Wyner</td>
<td>30%</td>
<td>11.07 ± 2.46</td>
<td>10.50 ± 3.03</td>
<td>6.64 ± 2.48</td>
<td>5.29 ± 2.41</td>
<td>9.50 ± 3.40</td>
</tr>
<tr>
<td>Mushroom</td>
<td>30%</td>
<td>0.49 ± 0.40</td>
<td>0.49 ± 0.41</td>
<td>0.44 ± 0.39</td>
<td>0.48 ± 0.29</td>
<td>1.02 ± 0.52</td>
</tr>
<tr>
<td>USPS-N</td>
<td>30%</td>
<td>7.27 ± 1.09</td>
<td>7.29 ± 1.19</td>
<td>6.62 ± 0.97</td>
<td>6.01 ± 0.84</td>
<td>4.35 ± 1.97</td>
</tr>
<tr>
<td>Adult</td>
<td>30%</td>
<td>15.79 ± 0.49</td>
<td>15.76 ± 0.46</td>
<td>15.72 ± 0.48</td>
<td>15.75 ± 0.47</td>
<td>15.87 ± 0.62</td>
</tr>
<tr>
<td>Web</td>
<td>30%</td>
<td>1.98 ± 0.18</td>
<td>1.97 ± 0.18</td>
<td>1.96 ± 0.17</td>
<td>1.95 ± 0.17</td>
<td>3.02 ± 0.23</td>
</tr>
</tbody>
</table>

Table 7: The test error (%) of RampSVM on selected datasets.