
Construction of non-convex polynomial loss functions for training a binary classifier with quantum annealing

Ryan Babbush
 Vasil Denchev
 Nan Ding
 Sergei Isakov
 Hartmut Neven

Google, 340 Main St, Venice, CA 90291

BABBUSH@FAS.HARVARD.EDU
 DENCHEV@GOOGLE.COM
 DINGNAN@GOOGLE.COM
 ISERGE@GOOGLE.COM
 NEVEN@GOOGLE.COM

Abstract

Quantum annealing is a heuristic quantum algorithm which exploits quantum resources to minimize an objective function embedded as the energy levels of a programmable physical system. To take advantage of a potential quantum advantage, one needs to be able to map the problem of interest to the native hardware with reasonably low overhead. Because experimental considerations constrain our objective function to take the form of a low degree PUBO (POLYNOMIAL UNCONSTRAINED BINARY OPTIMIZATION), we employ non-convex loss functions which are polynomial functions of the margin. We show that these loss functions are robust to label noise and provide a clear advantage over convex methods. These loss functions may also be useful for classical approaches as they compile to regularized risk expressions which can be evaluated in constant time with respect to the number of training examples.

1. Introduction

1.1. Quantum annealing

While it is well known that gate model quantum algorithms provide an exponential speedup over the best known classical approaches for some problems (Shor, 1997; Kitaev et al., 2002), we are still technologically far from the ability to construct a large scale quantum computer which can robustly implement such algorithms for nontrivial problem instances. By contrast, rapid advances in superconducting qubit technology (Barends et al., 2014) have provided a scalable platform for engineering medium-scale, controllable quantum systems at finite temperature. Such devices would

be able to implement a quantum version of simulated annealing (Kirkpatrick et al., 1983) known as quantum annealing (Kadowaki & Nishimori, 1998; Farhi et al., 2000; Santoro et al., 2002; Somma et al., 2008).

Because it is NP-HARD to determine the lowest energy configuration of a system of binary spins subject to controllable linear and quadratic energy terms (Barahona, 1982), the ability to engineer and cool such a system provides an approach to solving any optimization problem in the class NP. In general, we do not expect that any device can efficiently solve instances of NP-HARD problems in the worst case. However, there is evidence that quantum resources such as tunneling and entanglement are generic computational resources which may help to solve problem instances which would be otherwise intractable for classical solvers. For instance, quantum annealing allows disordered magnets to relax to states of higher magnetic susceptibility asymptotically faster than classical annealing (Brooke et al., 1999) and can solve certain oracular problems exponentially faster than any classical algorithm (Somma et al., 2012).

For the last few years, *D-Wave Systems* has been commercially manufacturing quantum annealing machines (Johnson et al., 2011). These machines are the subject of ongoing scientific investigations by several third parties which aim to characterize the extent to which the hardware utilizes quantum resources and whether a scaling advantage is apparent for any class of problems (Boixo et al., 2014).

1.2. Training under non-convex loss

The problem we consider in this work is the training of a linear binary classifier using noisy data (Bishop, 2006). We assume that the training data is provided as a matrix $\hat{\mathbf{x}} \in \mathbb{R}^{m \times n}$ with the m rows corresponding

to unique descriptor vectors containing n features. We are also provided with a vector of labels, $\mathbf{y} \in \{-1, 1\}^m$, which associate a binary classification with each feature vector. The training problem is to determine an optimal classifier $\mathbf{w} \in \mathbb{R}^n$ which predicts the data by classifying example i as $\text{sign}(\mathbf{w}^\top \mathbf{x}_i)$.

The classifier may be viewed as a hyperplane in feature space which divides data points into negative and positive classifications. In this space, the distance that example i falls from the classification hyperplane \mathbf{w} is referred to as the margin $\gamma_i \equiv y_i \mathbf{x}_i^\top \mathbf{w}$. Whereas a negative margin represents a classification opposite the training label, a positive margin represents a classification consistent with the training label. To cast training as an optimization problem we use the concept of a loss function which penalizes the classification of each example according to its margin (Bishop, 2006). Perhaps the simplest loss function is the 0-1 loss function which provides a correct classification with penalty 0 and an incorrect classification with penalty 1,

$$L_{01}(\gamma_i) \equiv \frac{1 - \text{sign}(\gamma_i)}{2}. \quad (1)$$

The training objective (known in machine learning as total empirical risk) is given as the mean loss over all examples in the training set. For instance, the 0-1 empirical risk function is

$$f_{01}(\mathbf{w}) \equiv \frac{1}{m} \sum_{i=0}^{m-1} L_{01}(\gamma_i). \quad (2)$$

Unfortunately, minimization of the 0-1 empirical risk function is known to be NP-HARD (Feldman et al., 2012). For this reason, most contemporary research focuses on convex loss functions which are provably efficient to optimize. However, in data with high label noise, this is an unacceptable compromise as the efficiency gained by convex minimization allows only for the efficient computation of a poor classifier (Manwani & Sastry, 2013). By contrast, training under non-convex loss functions is known to provide robust classifiers even when nearly half of the examples are mislabeled (Long & Servedio, 2010).

Objectives such as these, for which certain instances may require exponential time using classical heuristics, are ideal candidates for quantum annealing. In order to attempt non-convex risk minimization with quantum annealing in the near future, one must first *efficiently* compile the problem to a form compatible with quantum hardware. Due to engineering considerations, this usually means preparing the problem as an instance of QUBO (QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION). Previously, Denchev *et al.* in-

troduced a method for mapping non-convex loss training to QUBO (Denchev et al., 2012) for the purposes of solving on a quantum device. However, in that work, the number of variables required to accomplish the embedding was lower-bounded by the number of training examples. While clearly robust, this scheme seems impractical for medium-scale quantum annealers due to the large qubit overhead.

Here, we develop a different embedding in which the number of required variables is independent of the number of training examples. This is accomplished by deriving loss functions which are polynomial functions of the margins. We show that such loss functions give rise to empirical risk objectives expressible as PUBO. Compatibility with quantum hardware comes from the fact that any PUBO can be reduced to QUBO using a number of boolean ancilla variables that is at most $O(N^{2 \log k})$ where N is the number of logical variables and k is the order of the PUBO (Boros & Gruber, 2012). Coincidentally, this implies that the empirical risk objective associated with any polynomial loss function can be evaluated in an amount of time that does not depend on the number of training examples.

In particular, we investigate the use of third-order and sixth-order polynomial loss functions. The cubic loss function is chosen as $k = 3$ is the lowest order that gives us non-convexity. Polynomial loss has very different characteristics depending on the parity of k so we also investigate an even degree polynomial loss function. We forgo quartic loss in favor of sixth-order loss as the latter qualitatively fits 0-1 loss much better than the former. After deriving optimal forms of cubic loss and sixth-order loss we numerically investigate the properties of these loss functions to show robustness to label noise. Finally, we demonstrate an explicit mapping of any polynomial risk objective to a tensor representing an instance of PUBO that is easily compiled to quantum hardware.

2. Cubic loss

In this section we derive an approximate embedding of 0-1 risk under ℓ_2 -norm regularization as a cubic function of the weights. We begin by considering the general forms of the cubic loss and cubic risk functions,

$$L_3(\gamma_i) = \alpha_0 + \alpha_1 \gamma_i + \alpha_2 \gamma_i^2 + \alpha_3 \gamma_i^3 \quad (3)$$

$$f_3(\mathbf{w}) = \frac{1}{m} \sum_{i=0}^{m-1} L_3(\gamma_i). \quad (4)$$

Thus, the embedding problem is to choose the optimal $\boldsymbol{\alpha} \in \mathbb{R}^4$ so that $f_3(\mathbf{w})$ best approximates $f_{01}(\mathbf{w})$. To accomplish this we consider the ℓ_2 -norm between 0-1

risk and cubic risk,

$$\boldsymbol{\alpha}^* = \operatorname{argmin} \left\{ \int P(\mathbf{w}) [f_{01}(\mathbf{w}) - f_3(\mathbf{w})]^2 d\mathbf{w} \right\}. \quad (5)$$

Here, $P(\mathbf{w})$ is the prior distribution of the weights. If we incorporate an ℓ_2 -norm regularizer, $\Omega_2(\mathbf{w})$, into our ultimate training objective, $E(\mathbf{w})$, i.e.

$$\Omega_2(\mathbf{w}) = \frac{\lambda_2}{2} \mathbf{w}^\top \mathbf{w} \quad (6)$$

$$E(\mathbf{w}) = f(\mathbf{w}) + \Omega_2(\mathbf{w}), \quad (7)$$

then we are provided with a Gaussian prior on the weights (Rennie, 2003) taking the form,

$$P(w_i) = \sqrt{\frac{\lambda_2}{2\pi}} e^{-\lambda_2 w_i^2 / 2}. \quad (8)$$

Immediately, we see that for the optimal solution $\alpha_2 \rightarrow 0$ since 0-1 loss is an odd function and the least squares residual is weighed over a symmetric function (the Gaussian prior). Furthermore, we can ignore α_0 and the constant factor of $\frac{1}{2}$ in $L_{01}(\gamma_i)$ as these constants are irrelevant for the training problem. With this in mind, we expand the empirical risk functions under the integral in the embedding problem as,

$$\int P(\mathbf{w}) \left(\sum_{i=0}^{m-1} \frac{\operatorname{sign}(\gamma_i)}{2} + \alpha_1 \gamma_i + \alpha_3 \gamma_i^3 \right)^2 d\mathbf{w}. \quad (9)$$

Thus,

$$\boldsymbol{\alpha}^* = \operatorname{argmin} \left\{ \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \int P(\boldsymbol{\gamma}) F_{ij}(\boldsymbol{\gamma}) d\boldsymbol{\gamma} \right\} \quad (10)$$

where

$$\begin{aligned} F_{ij}(\boldsymbol{\gamma}) &\equiv \frac{\alpha_1}{2} [\gamma_i \operatorname{sign}(\gamma_j) + \gamma_j \operatorname{sign}(\gamma_i)] \quad (11) \\ &+ \frac{\alpha_3}{2} [\gamma_i^3 \operatorname{sign}(\gamma_j) + \gamma_j^3 \operatorname{sign}(\gamma_i)] \\ &+ \alpha_1^2 \gamma_i \gamma_j + \alpha_1 \alpha_3 (\gamma_i^3 \gamma_j + \gamma_j^3 \gamma_i) + \alpha_3^2 \gamma_i^3 \gamma_j^3. \end{aligned}$$

Without loss of generality, we may assume that $P(\boldsymbol{\gamma})$ is a multinormal distribution centered at zero with a covariance matrix,

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{\lambda_2} (\hat{\boldsymbol{x}}^\top \hat{\boldsymbol{x}}) \odot (\mathbf{y} \mathbf{y}^\top) \quad (12)$$

where \odot implies element-wise matrix multiplication (i.e. the Hadamard product). The multinormal distribution occurs because the margins arise as the result of the training examples being projected by classifiers drawn from the prior distribution given by $P(\mathbf{w})$.

Since each weight is normally distributed with zero mean and variance λ_2^{-2} in the prior, the distribution of margins associated with training example i will be a Gaussian with zero mean and variance,

$$\sigma_i^2 = \frac{1}{n} \sum_{j=0}^{n-1} \left(\frac{x_{ij}}{\lambda_2} \right)^2. \quad (13)$$

Because each linear combination of the elements of the margin vector is also normally distributed, we have a multinormal distribution. This is true regardless of the number of features or any particular qualities of the training data.

Accordingly, if we wished to scale \mathbf{w} to a range which contains \mathbf{w}^* with a likelihood in the r^{th} standard deviation of the prior then we should make $\mathbf{w} \in \left[-\frac{r}{\sqrt{\lambda_2 m}}, \frac{r}{\sqrt{\lambda_2 m}} \right]^n$. However, making r too large would be problematic because this could allow the cubic term to dominate the quadratic regularizer. This necessitates a cutoff on the maximum weight value to ensure that unbounded cubic losses associated with large negative margins do not overcome the regularizer. In practice, r would need to be selected as a hyperparameter.

Since the integrand has only two point correlation functions, we can integrate over the marginal distribution of γ_i and γ_j which is a binormal distribution with covariance,

$$\hat{\boldsymbol{\Sigma}}_{ij} = \frac{1}{\lambda_2} \begin{pmatrix} \mathbf{x}_i^\top \mathbf{x}_i & y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ y_j y_i \mathbf{x}_j^\top \mathbf{x}_i & \mathbf{x}_j^\top \mathbf{x}_j \end{pmatrix} \quad (14)$$

$$= \begin{pmatrix} \sigma_i^2 & \rho_{ij} \sigma_i \sigma_j \\ \rho_{ij} \sigma_j \sigma_i & \sigma_j^2 \end{pmatrix}. \quad (15)$$

We can analytically evaluate the double integral,

$$\boldsymbol{\alpha}^* = \operatorname{argmin} \left\{ \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} I_{ij} \right\} \quad (16)$$

$$\begin{aligned} I_{ij} &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P_{ij}(\boldsymbol{\gamma}) F_{ij}(\boldsymbol{\gamma}) d\boldsymbol{\gamma} d\boldsymbol{\gamma} \quad (17) \\ &= \underbrace{\frac{\rho_{ij}(\sigma_i + \sigma_j)}{\sqrt{2\pi}} \alpha_1}_{t_0} + \underbrace{\frac{\rho_{ij}(3 - \rho_{ij}^2)(\sigma_i^3 + \sigma_j^3)}{\sqrt{2\pi}} \alpha_3}_{t_1} \\ &\quad + 3 \underbrace{\rho_{ij} \sigma_i \sigma_j (\sigma_i^2 + \sigma_j^2)}_{t_2} \alpha_1 \alpha_3 + \underbrace{\rho_{ij} \sigma_i \sigma_j \alpha_1^2}_{t_3} \\ &\quad + 3 \underbrace{\rho_{ij} (3 + 2\rho_{ij}^2) \sigma_i^3 \sigma_j^3 \alpha_3^2}_{t_4} \\ &= t_0 \alpha_1 + t_1 \alpha_3 + t_2 \alpha_1 \alpha_3 + t_3 \alpha_1^2 + t_4 \alpha_3^2 \end{aligned}$$

With the integral in closed form, we obtain the argmin

using simple algebra by solving,

$$\nabla \left(\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} I_{ij} \right) = 0. \quad (18)$$

The analytical coefficients for a single set of training examples are,

$$\alpha_1^* = \frac{2t_0t_4 - t_1t_2}{t_2^2 - 4t_3t_4} \quad \alpha_3^* = \frac{2t_1t_3 - t_0t_2}{t_2^2 - 4t_3t_4}. \quad (19)$$

Thus, for the full training set we must sum together the t values from each set of (i, j) before plugging values into the expression for α_1 and α_3 . We note that the computation required to obtain these coefficients is $O(m^2)$. Figure 1 shows several cubic loss function fits associated with various real data sets from the UCI Machine Learning Repository.

The coefficients above are analytic and optimal for embedding 0-1 risk. However, it is instructive to explain what would happen if we had chosen to fit the loss function instead of the objective function. This would have produced a far simpler embedding problem¹,

$$\alpha^* = \operatorname{argmin} \left\{ \int_{-\infty}^{\infty} P^*(\gamma) [L_{01}(\gamma) - L_3(\gamma)]^2 d\gamma \right\} \quad (20)$$

where

$$P^*(\gamma) \equiv \frac{1}{\sigma\sqrt{2\pi}} e^{-\gamma^2/2\sigma^2}, \quad \sigma^2 = \frac{1}{\lambda_2 m} \operatorname{tr}[\hat{\mathbf{x}}^\top \hat{\mathbf{x}}]. \quad (21)$$

This time the integral is trivial to evaluate,

$$\begin{aligned} I^* &= \int_{-\infty}^{\infty} G^*(\gamma, \alpha) [L_{01}(\gamma) - L_3(\gamma)]^2 d\gamma \quad (22) \\ &= \sqrt{\frac{2}{\pi}} \sigma \alpha_1 + \sigma^2 \alpha_1^2 + 2\sqrt{\frac{2}{\pi}} \sigma^3 \alpha_3 \\ &\quad + 6\sigma^4 \alpha_1 \alpha_3 + 15\sigma^6 \alpha_3^2. \end{aligned}$$

As before, convexity guarantees that ∇I will have exactly one real root which we find analytically,

$$\alpha_1^* = -\frac{3}{2\sqrt{2\pi}\sigma} \quad \alpha_3^* = \frac{1}{6\sqrt{2\pi}\sigma^3}. \quad (23)$$

These result seems much simpler than when we embed the entire risk function but they are obviously less useful. However, if we make the assumption that $\sigma_i = \sigma_j = \sigma$ then, $\alpha_1^* = \alpha_1^*$ and $\alpha_3^* = \alpha_3^*$.

In Figure 2 we study the performance of our embedded loss function by exactly enumerating the solution space produced by small synthetic data sets. These data sets

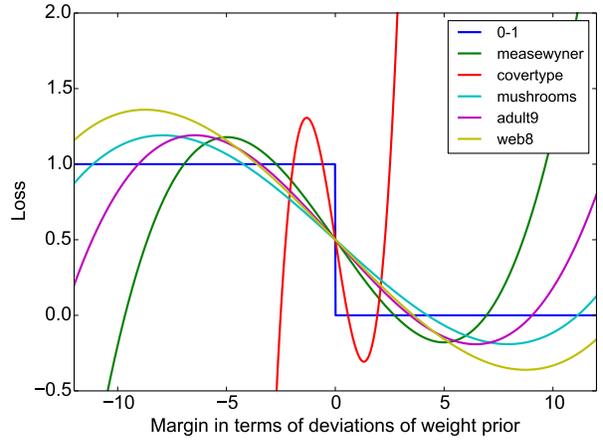


Figure 1:

Cubic loss fits for a variety of real data sets. Due to the different properties of their correlation matrices each set is associated with unique cubic loss coefficients.

were produced by randomly generating classifiers with weights drawn from a normal distribution and then using that classifier to label feature vectors with features drawn from a uniform distribution. Symmetric label errors are then manually injected at random. A maximum weight cutoff is imposed at the second standard deviation of the weight prior. Further numerical analysis of cubic loss on synthetic data sets is included in the Appendix.

While the cubic loss embedding is somewhat noisy in the sense that it does not perfectly approximate 0-1 loss, it is clearly robust in the sense that test error does not depend strongly on label error for up to 45% label noise. This remains true whether we consider the best fifty states embedded in cubic loss or only the absolute ground state. These results indicate that cubic loss has an advantage over convex methods when data is known to contain substantial label noise.

3. Sixth-order loss

One potentially unattractive feature of the cubic loss function is that it is necessary to fix the scale of the weights as a hyperparameter. Since we intend to encode our objective function as QUBO for quantum annealing, we will need to choose a maximum weight. While one can prove that the optimal classifier will have weights in the interval $\left[-\frac{1}{\sqrt{\lambda_2}}, \frac{1}{\sqrt{\lambda_2}}\right]$, such a large range is potentially problematic for regularized cubic risk as the loss associated with large negative margins tends towards negative infinity faster than the ℓ_2 regularizer can penalize the large weights which would produce such margins.

¹Note the difference between α^* and α^* .

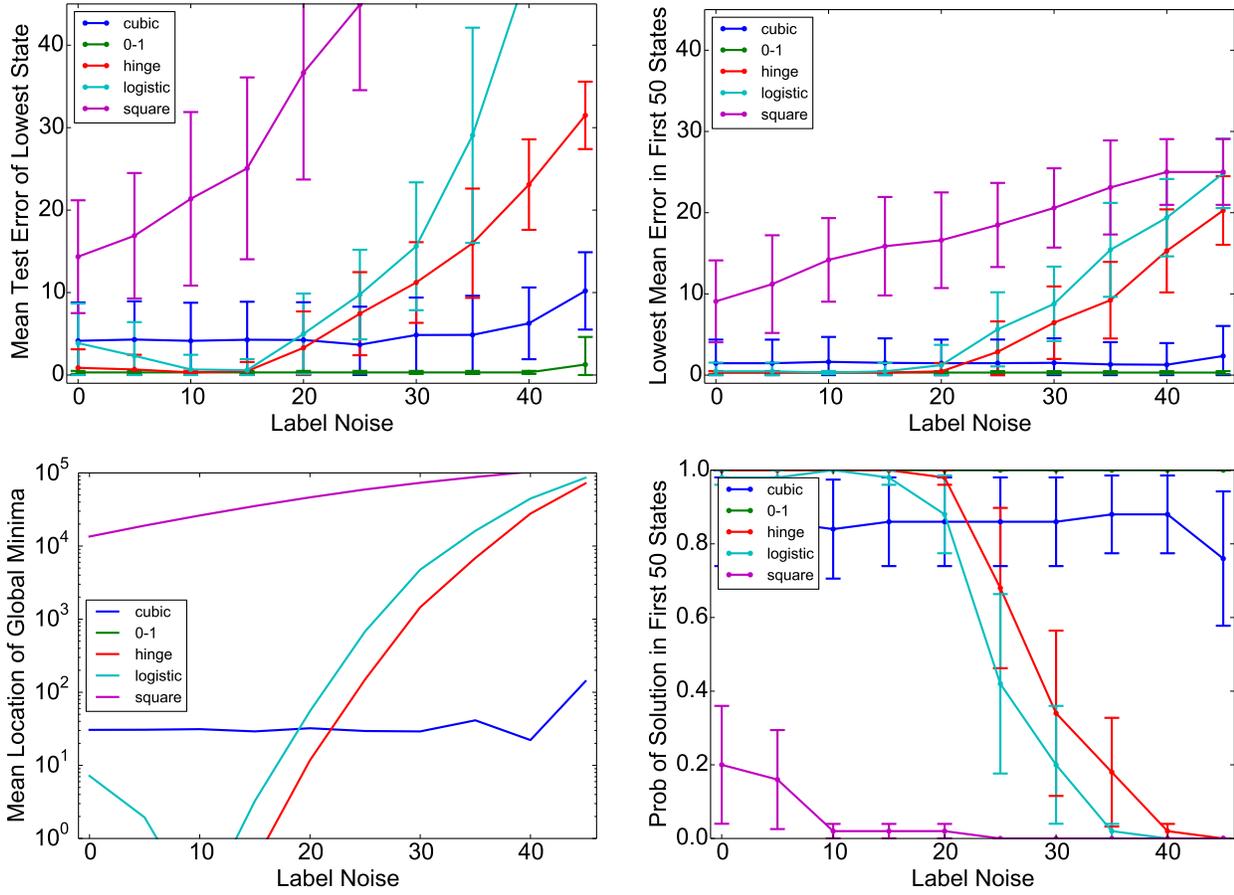


Figure 2: Performance of cubic loss on synthetic data sets with 10^4 examples, 9 features and a bit depth of 2. We exactly enumerate all fixed bit depth classifiers and evaluate the empirical risk under various loss functions. Error bars are obtained by repeating the experiment on fifty data sets. The upper left plot shows the error in the lowest objective state and the upper right plot shows the error in the best state of the lowest fifty. The lower left plot shows mean position of the global minima in the eigenspectra of the various objective functions. The lower right plot shows the probability of the global minima being in the first fifty states. As we can see, the global minima remains very near the bottom of the eigenspectra for cubic loss, regardless of label noise.

Alternatively, one might consider using a polynomial loss function of even degree as such loss functions will not diverge to negative infinity for large negative margins. In order to do this, we must fix the highest order term in the loss function as a hyperparameter. This is because 0-1 loss is an odd function so if we attempt to embed 0-1 risk in an even degree polynomial, the even terms will vanish. Accordingly, we turn our attention to the sixth-order loss and empirical risk functions,

$$L_6(\gamma_i) \equiv \omega \gamma_i^6 + \sum_{k=1}^5 \beta_k \gamma_i^k \quad (24)$$

$$f_6(\mathbf{w}) \equiv \frac{1}{m} \sum_{i=0}^{m-1} L_3(\gamma_i). \quad (25)$$

Here, ω is taken to be a hyperparameter. We will solve for the values of β . In the case of the cubic loss function, we used the weight prior imposed by ℓ_2 -norm regularization and data set covariance to derive a margin prior which was used for embedding. However, this is unnecessary for the sixth-order loss function as ω provides a very simple prior on the margins,

$$P(\gamma) = \frac{\omega^{1/6}}{2\Gamma(7/6)} e^{-\omega\gamma^6} \quad (26)$$

where Γ is the standard gamma function. With this definition, the embedding problem becomes

$$\beta^* = \operatorname{argmin} \left\{ \int P(\gamma) [L_{01}(\gamma) - L_6(\gamma)]^2 d\gamma \right\}. \quad (27)$$

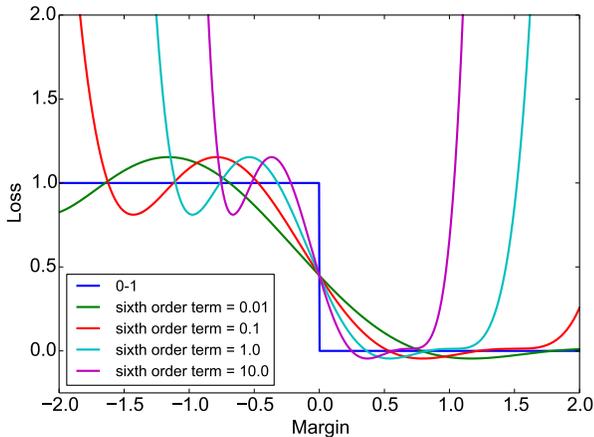


Figure 3: The sixth-order loss function at various values of the fixed sixth-order coefficient, ω . This coefficient is taken to be a hyperparameter.

Whereas we chose α for cubic loss by fitting the empirical risk function, we choose β for sixth-order loss by fitting the loss function directly. Since the sixth-order loss function is already parameterized in terms of a hyperparameter, there is nothing to gain by devising a more elaborate fit based on empirical risk. After evaluating I_6 , the integral in Eq. 27, we can obtain β^* by solving, $\nabla I_6 = 0$. The optimal values of β are included in the Appendix. Figure 3 shows the sixth-order loss function for various values of ω .

Figure 4 (on the next page) shows the performance of the sixth-order loss function on selected data sets from the UCI Machine Learning Repository. To stand-in for a quantum annealer, we optimized the sixth-order objective function using a simulated annealing routine which was run for over one hundred thousand CPU hours. In addition to standard convex methods, we compare to 0-1 loss optimized using the same simulated annealing code run with the same number of restarts and variable updates as were used to optimize sixth-order loss. We also include the “q-loss” results from (Denchev et al., 2012) which were obtained for that work using another metaheuristic algorithm (Tabu search). Details regarding the 10-fold cross validation procedure are reported in the Appendix.

We find that for all real data sets, the sixth-order loss function outperforms all tested convex loss functions and performs similarly to the non-convex methods. The first two data sets shown in Figure 4 are synthetic sets designed to break convex loss functions devised by Long and Servedio (Long & Servedio, 2010) and Mease and Wyner (Mease & Wyner, 2008). The sixth-order loss performs poorly on the Long-Servedio set because the data set is designed so that the op-

timal solution has only extremely large margins and extremely small margins. The large margins dominate the risk minimization due to the steep walls of the sixth-order function and this forces all of the smaller margins very near zero where the sixth-order function is almost linear. We believe that the particular pathological behavior which leads to the poor performance of sixth-order loss on the Long-Servedio set is unlikely to occur in real data.

Figure 4 shows that the sixth-order loss function outperforms even the other non-convex methods on three of the four real data sets. However, we attribute the suboptimal q-loss and 0-1 loss results to a failure of the selected optimization routines rather than to a deficiency of the actual training objectives. One reason this seems likely is because sixth-order loss outperforms the other non-convex functions most significantly on web8 (the real data set with the greatest number of features) but losses to q-loss and 0-1 loss on covtype (the real data set with the fewest number of features). A comprehensive summary of the data sets is included in the Appendix. While non-convex, the sixth-order loss objective appears to be somewhat easier to optimize as a consequence of being significantly smoother than either the 0-1 loss or q-loss objective.

4. Explicit tensor construction

In this section we show how to represent any regularized risk objective using a polynomial loss function as PUBO. We first introduce a fixed bit-depth approximation. More substantially, we represent variables using a fixed-point representation as floating-point representations require a non-polynomial function of the bits. Using d bits per feature, our encoding will require a total of $N = nd$ bits. The binary state vector $\mathbf{q} \in \mathbb{B}^N$ encodes the weight vector \mathbf{w} ,

$$\mathbf{w}[i] \equiv \zeta \mathbf{q}[id] - \zeta \sum_{j=1}^{d-1} \left(\frac{1}{2}\right)^j \mathbf{q}[id + j] \quad (28)$$

where $\zeta \in \mathbb{R}$ determines the weight scale so that $\mathbf{w} \in (-\zeta, \zeta]^n$. Furthermore, we define a binary coefficient matrix, $\hat{\mathbf{k}} \in \mathbb{R}^{n \times N}$,

$$\hat{\mathbf{k}} \equiv \mathbf{I}^{n \times n} \otimes \left\langle \zeta, -\frac{\zeta}{2}, -\frac{\zeta}{4}, \dots, \frac{\zeta}{2^{1-d}} \right\rangle \quad (29)$$

where $\mathbf{I}^{n \times n} \otimes$ indicates a Kronecker product by an $n \times n$ identity matrix. This “tiles” the binary weight sequence into a stair-step pattern down the diagonal; e.g. if $n = 3$, $d = 2$ and $\zeta = 1$,

$$\hat{\mathbf{k}} = \begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{1}{2} \end{pmatrix}. \quad (30)$$

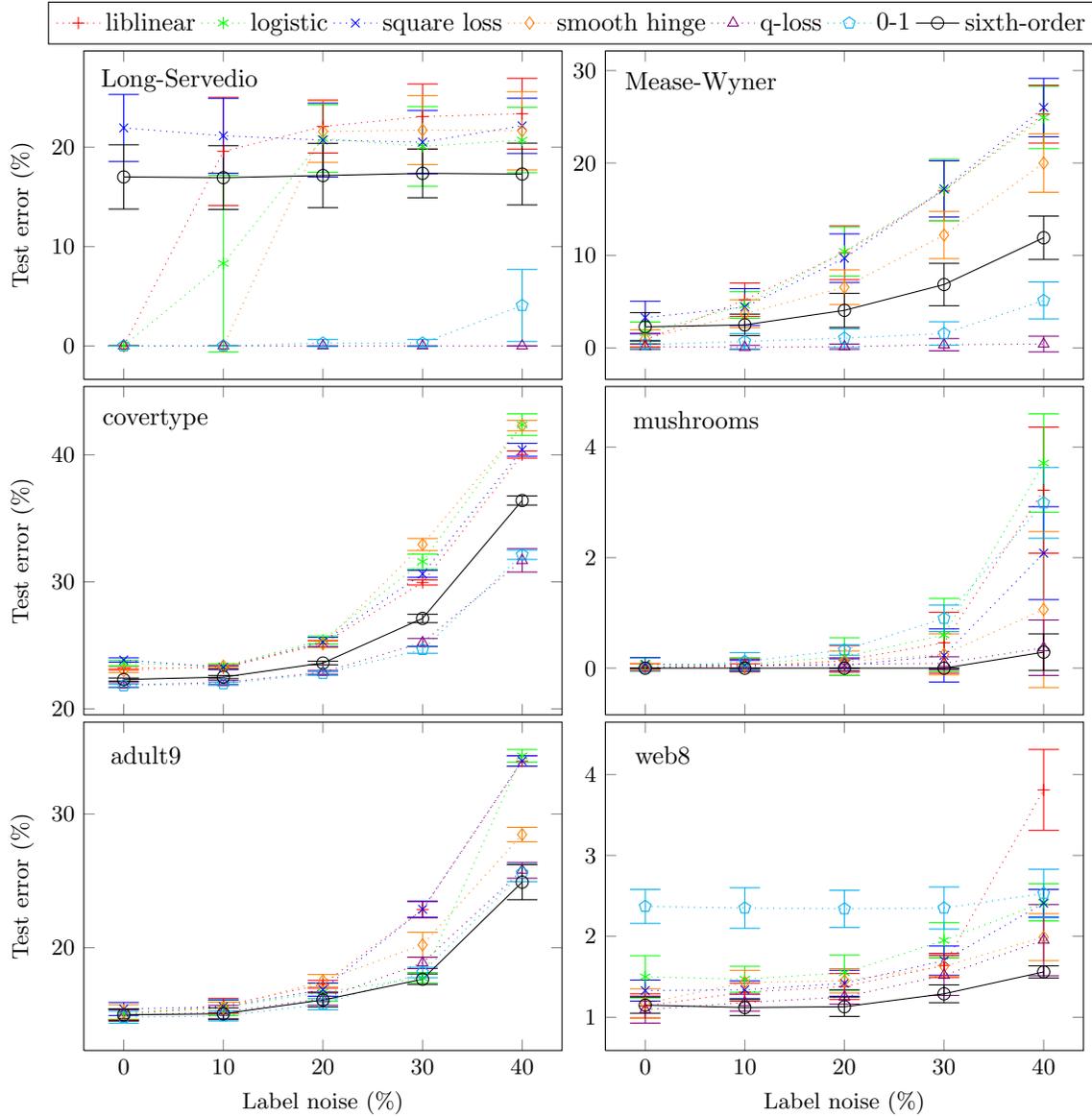


Figure 4: Test error versus label noise for 7 methods on 2 synthetic data sets (Long-Servedio and Mease-Wyner) and 4 real data sets from the UCI repository. Error bars are obtained from 10-fold cross-validation with the hyperparameters recorded in the Appendix. As a stand-in for quantum annealing, a classical simulated annealing routine was used to optimize the sixth-order objective function and the 0-1 objective function. For each training cut at each selection of hyperparameters, we kept 50 classifiers having the lowest objective values of all states encountered. We computed validation error as the lowest of validation error produced by these 50 classifiers. This procedure is used for both 0-1 loss and sixth-order loss. Test error was obtained using the classifier of lowest validation error. This strategy is realistic as we expect that a quantum annealer will sample the lowest energy states as opposed to giving us only the global minima. q-loss was optimized using Tabu search in (Denchev et al., 2012). We see that sixth-order loss outperforms the convex methods on every data set except for Long-Servedio and performs similarly to other non-convex methods on the other five sets.

We do this because later on, it will be useful to think of \mathbf{w} as a linear mapping of \mathbf{q} given as $\mathbf{w} = \hat{\mathbf{k}}\mathbf{q}$. In general, any PUBO of degree k can be expressed as,

$$E(\mathbf{q}) = \mathbf{v}^\top \mathbf{q}^{\otimes k} \quad (31)$$

where $\mathbf{v} \in \mathbb{R}^{N^k}$ is a k -fold tensor and $\mathbf{q}^{\otimes k}$ represents the k^{th} tensor power of \mathbf{q} . We now show how to obtain this embedding for a cubic loss function but do so in a way that is trivially extended to orders less than or greater than cubic. In terms of continuous weights the empirical risk objective may be expressed as,

$$\begin{aligned} f(\mathbf{w}) &= \frac{1}{m} \sum_{i=0}^{m-1} \alpha_1 y_i \mathbf{x}_i^\top \mathbf{w} + \alpha_2 (\mathbf{x}_i^\top \mathbf{w})^2 + \alpha_3 (y_i \mathbf{x}_i^\top \mathbf{w})^3 \\ &= \underbrace{\left(\frac{\alpha_1}{m} \sum_{i=0}^{m-1} y_i \mathbf{x}_i^\top \right)}_{\boldsymbol{\varphi}_1^\top} \mathbf{w} + \underbrace{\left(\frac{\alpha_2}{m} \sum_{i=0}^{m-1} (\mathbf{x}_i^{\otimes 2})^\top \right)}_{\boldsymbol{\varphi}_2^\top} \mathbf{w}^{\otimes 2} \\ &+ \underbrace{\left(\frac{\alpha_3}{m} \sum_{i=0}^{m-1} (y_i \mathbf{x}_i^{\otimes 3})^\top \right)}_{\boldsymbol{\varphi}_3^\top} \mathbf{w}^{\otimes 3} = \sum_{j=1}^3 \boldsymbol{\varphi}_j^\top \mathbf{w}^{\otimes j} \end{aligned} \quad (32)$$

where

$$\boldsymbol{\varphi}_j = \frac{\alpha_j}{m} \sum_{i=0}^{m-1} (y_i \mathbf{x}_i)^{\otimes j}. \quad (33)$$

Using tensor notation, ℓ_2 -norm regularization is

$$\Omega_2(\mathbf{w}) = \frac{\lambda_2}{2} (\mathbf{1}^{n^2})^\top \mathbf{w}^{\otimes 2}, \quad (34)$$

where $\mathbf{1}^{n^2}$ denotes a vector of all ones with length n^2 . We now use $\hat{\mathbf{k}}$ to expand the binary variable tensor,

$$\begin{aligned} E(\mathbf{q}) &= f(\mathbf{w}) + \Omega_2(\mathbf{w}) \\ &= \frac{\lambda_2}{2} (\mathbf{1}^{n^2})^\top (\hat{\mathbf{k}}\mathbf{q})^{\otimes 2} + \sum_{j=1}^3 \boldsymbol{\varphi}_j^\top (\hat{\mathbf{k}}\mathbf{q})^{\otimes j}. \end{aligned} \quad (35)$$

This expression implies the form of \mathbf{v} ,

$$\mathbf{v} = \frac{\lambda_2}{2} \mathbf{1}^{n^2} \otimes \hat{\mathbf{k}}^{\otimes 2} + \sum_{j=1}^3 \boldsymbol{\varphi}_j \otimes \hat{\mathbf{k}}^{\otimes j}. \quad (36)$$

We slightly abuse notation in our definition of \mathbf{v} by “adding” together tensors of different rank. To accomplish this the tensor of lower rank should be placed in a tensor having the same rank as the larger tensor by setting additional tensor indices equal to a lower tensor index. For instance, the element corresponding to (i, j) in a tensor of rank two could be placed in a tensor of rank three at (i, j, i) or (i, j, j) . We note that it is necessary to *first* convert to binary and then combine the

three terms; doing things the other way would introduce complications due to the fact that $w_i^r \neq w_i \forall i, r$ whereas $q_i^r = q_i \forall i, r$. Finally, we note that constructing $\boldsymbol{\varphi}_3^\top \otimes \hat{\mathbf{k}}^{\otimes 3}$ is the most computationally expensive part of this entire procedure taking $O(n^3 d^3 m)$ time.

This 3-fold tensor can be reduced to a QUBO matrix using ancillae. The optimal reduction is trivial using the tools developed in (Babbush et al., 2013). In Appendix B of that paper, it shown that the number of ancillae which are required to collapse a fully connected cubic to 2-local is upper bounded by $\frac{N^2}{4}$. Again, the general bound for the quadratization of a PUBO of degree k is $O(N^{2 \log k})$ (Boros & Gruber, 2012). This bound suggests that unlike prior encodings, the number of ancillae required is entirely independent of the number of training examples. We point out that the tensor form of this problem may be evaluated in a time that does not depend on the number of training examples.

5. Conclusion

We have introduced two unusual loss functions: the cubic loss function and the sixth-order loss function. Both losses are non-convex and show clear evidence of robustness to label noise. While superior to classically tractable convex training methods, both loss functions are highly parameterized and represent less than perfect approximations to 0-1 loss. Prima facie, this suggests that more popular non-convex loss functions, e.g. sigmoid loss, may be more reliable (or at least more straightforward) in some respects.

However, training under non-convex loss is formally NP-Hard and in order to obtain satisfactory solutions to such optimization problems, heuristic algorithms must query the objective function many times. Often, the quality of the eventual solution depends on the number of queries the heuristic is allowed. The fact that the polynomial loss functions may be compiled so that each query to the objective is independent of the number of training examples suggests that these loss functions may be more compatible with heuristic optimization routines. This same property ensures that these loss functions can be compiled to a Hamiltonian suitable for quantum annealing using a reasonable number of qubits (estimates of resources requirements for various example problems are included in the Appendix). This efficient embedding in quantum hardware makes binary classification under non-convex polynomial loss a promising target problem to accelerate using a quantum annealing machine.

6. Appendix

6.1. ℓ_0 -norm regularization

While ℓ_0 programming is well known to be NP-Hard, we believe that quantum annealing may allow us to obtain satisfactory minima in many instances. ℓ_0 -norm regularization is often used to train classifiers that are particularly efficient in terms of the number of features required for classification. For the situation in which we would like to train a classifier with binary weights, the regularization function is trivial,

$$\Omega_0(\mathbf{q}) = \lambda_0 \sum_{i=0}^{n-1} q_i. \quad (37)$$

For multiple bit depth weights, ℓ_0 -norm regularization will require a modest number of ancilla qubits (one for each feature). Using our notation, the regularizer is

$$\Omega_0(\mathbf{q}) = \sum_{j=0}^{n-1} \left(\lambda_0 \mathbf{q}[N+j] + \phi (1 - \mathbf{q}[N+j]) \sum_{k=0}^{d-1} \mathbf{q}[jd+k] \right). \quad (38)$$

Minimizing $\Omega(\mathbf{q})$ causes the ancillae $\mathbf{q}[N+j]$ to act as indicator bits, each of which is 1 if and only if $w_j \neq 0$ and is 0 otherwise. This is achieved by summing the binary variables that take part in a weight variable. Correctness comes from the fact that the binary representation of $w_j = 0$ is when all bits corresponding to that weight are 0. Thus, if even a single bit from weight w_i is on, the objective will either incur a penalty of ϕ or will set the ancilla to 0 so as to obtain a penalty of λ_0 instead. Thus, as long as ϕ is sufficiently larger than λ_0 , this function enforces ℓ_0 -norm regularization with weight of λ_0 . This regularizer may be combined with the empirical risk function described previously.

6.2. Sixth-order loss coefficients

$$\beta_0 = \frac{1}{3} + \frac{343 \left(125\pi^{3/2} - 864 \Gamma\left(\frac{11}{6}\right)^3 \right)}{1296 \left(343 \Gamma\left(\frac{11}{6}\right)^3 + 750 \Gamma\left(\frac{13}{6}\right)^3 \right) - 300125\pi^{3/2}} \quad (39)$$

$$\beta_1 = -\frac{35\sqrt[3]{\omega} \left(245 \left(1 + 3 \cdot 2^{2/3} \right) \pi \Gamma\left(\frac{4}{3}\right) + 36 \left(49 \Gamma\left(\frac{11}{6}\right) \left(\sqrt{\pi} - 3 \Gamma\left(\frac{5}{3}\right) \Gamma\left(\frac{11}{6}\right) \right) - 60 \Gamma\left(\frac{13}{6}\right)^2 \right) \right)}{3 \left(-\frac{222950\pi^{3/2}}{9} + 98784 \Gamma\left(\frac{11}{6}\right)^3 + 43200 \Gamma\left(\frac{13}{6}\right)^3 \right)} \quad (40)$$

$$\beta_2 = \frac{2940\sqrt{\pi}\sqrt[3]{\omega} \left(25\sqrt{\pi} \Gamma\left(\frac{13}{6}\right) - 42 \Gamma\left(\frac{11}{6}\right)^2 \right)}{300125\pi^{3/2} - 1296 \left(343 \Gamma\left(\frac{11}{6}\right)^3 + 750 \Gamma\left(\frac{13}{6}\right)^3 \right)} \quad (41)$$

$$\beta_3 = \frac{3675\sqrt{\pi}\sqrt{\omega} \left(7\sqrt{\pi} + 63 \left(\sqrt[3]{2} - 1 \right) \Gamma\left(\frac{5}{3}\right) \Gamma\left(\frac{11}{6}\right) + 18 \left(2^{2/3} - 3 \right) \Gamma\left(\frac{4}{3}\right) \Gamma\left(\frac{13}{6}\right) \right)}{111475\pi^{3/2} - 1296 \left(343 \Gamma\left(\frac{11}{6}\right)^3 + 150 \Gamma\left(\frac{13}{6}\right)^3 \right)} \quad (42)$$

$$\beta_4 = \frac{2100\sqrt{\pi}\omega^{2/3} \left(49\sqrt{\pi} \Gamma\left(\frac{11}{6}\right) - 60 \Gamma\left(\frac{13}{6}\right)^2 \right)}{432 \left(343 \Gamma\left(\frac{11}{6}\right)^3 + 750 \Gamma\left(\frac{13}{6}\right)^3 \right) - \frac{300125\pi^{3/2}}{3}} \quad (43)$$

$$\beta_5 = \frac{7\omega^{5/6} \left(-7056 \Gamma\left(\frac{11}{6}\right)^2 + 1225 \left(3\sqrt[3]{2} - 1 \right) \pi \Gamma\left(\frac{5}{3}\right) + 600 \Gamma\left(\frac{13}{6}\right) \left(7\sqrt{\pi} - 18 \Gamma\left(\frac{4}{3}\right) \Gamma\left(\frac{13}{6}\right) \right) \right)}{-\frac{222950\pi^{3/2}}{9} + 98784 \Gamma\left(\frac{11}{6}\right)^3 + 43200 \Gamma\left(\frac{13}{6}\right)^3} \quad (44)$$

6.3. Convergence of cubic loss function

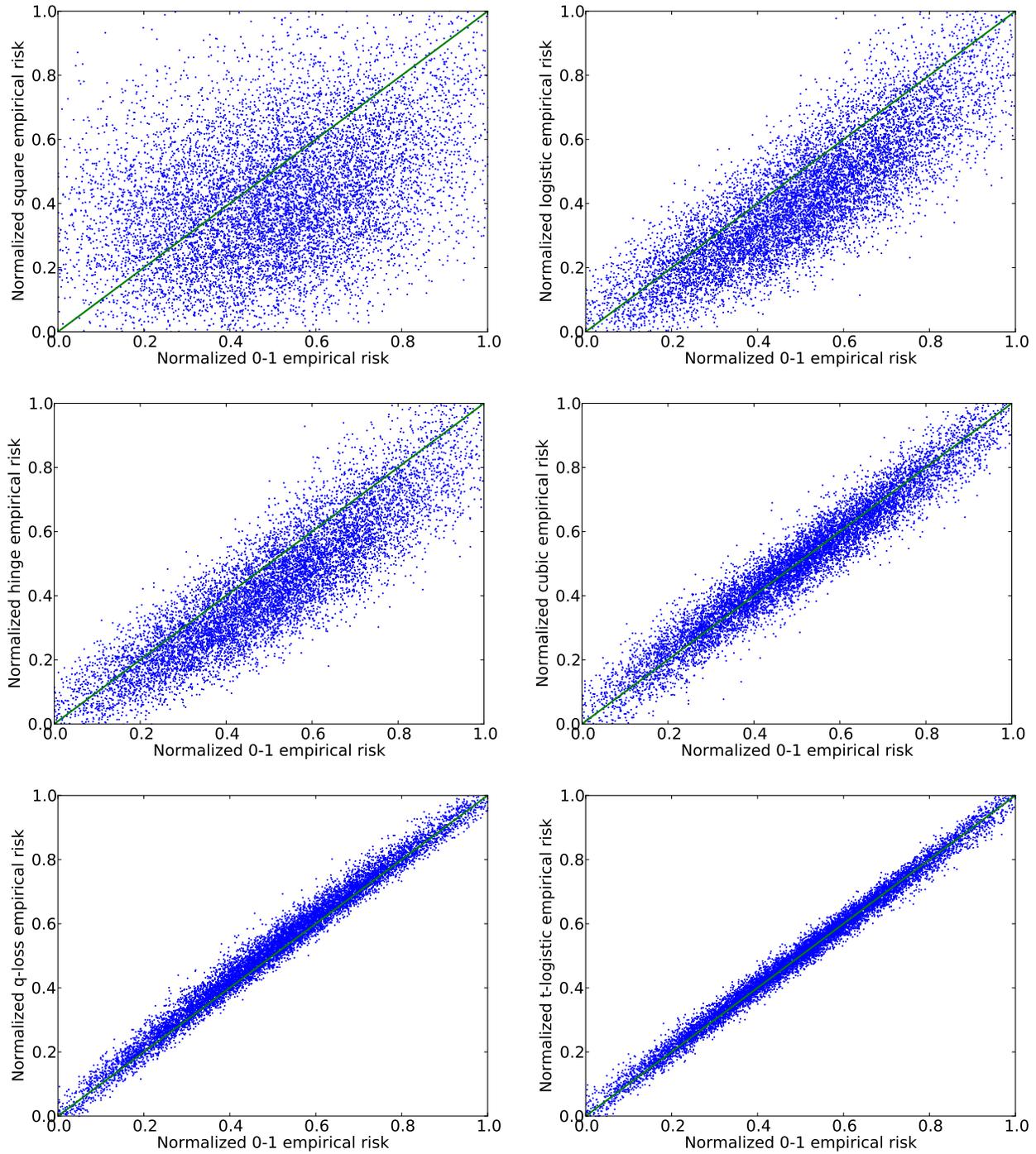


Figure 5: Correlations between the total empirical risk of 10^4 randomly selected states using various loss functions over 10^4 training examples from the adult9 data set. The empirical risk values of each state have been uniformly shifted and rescaled to be in between 0 and 1. As we can see, the correlations between the convex loss functions and 0-1 loss are strictly worse than the correlation between cubic loss and 0-1 loss.

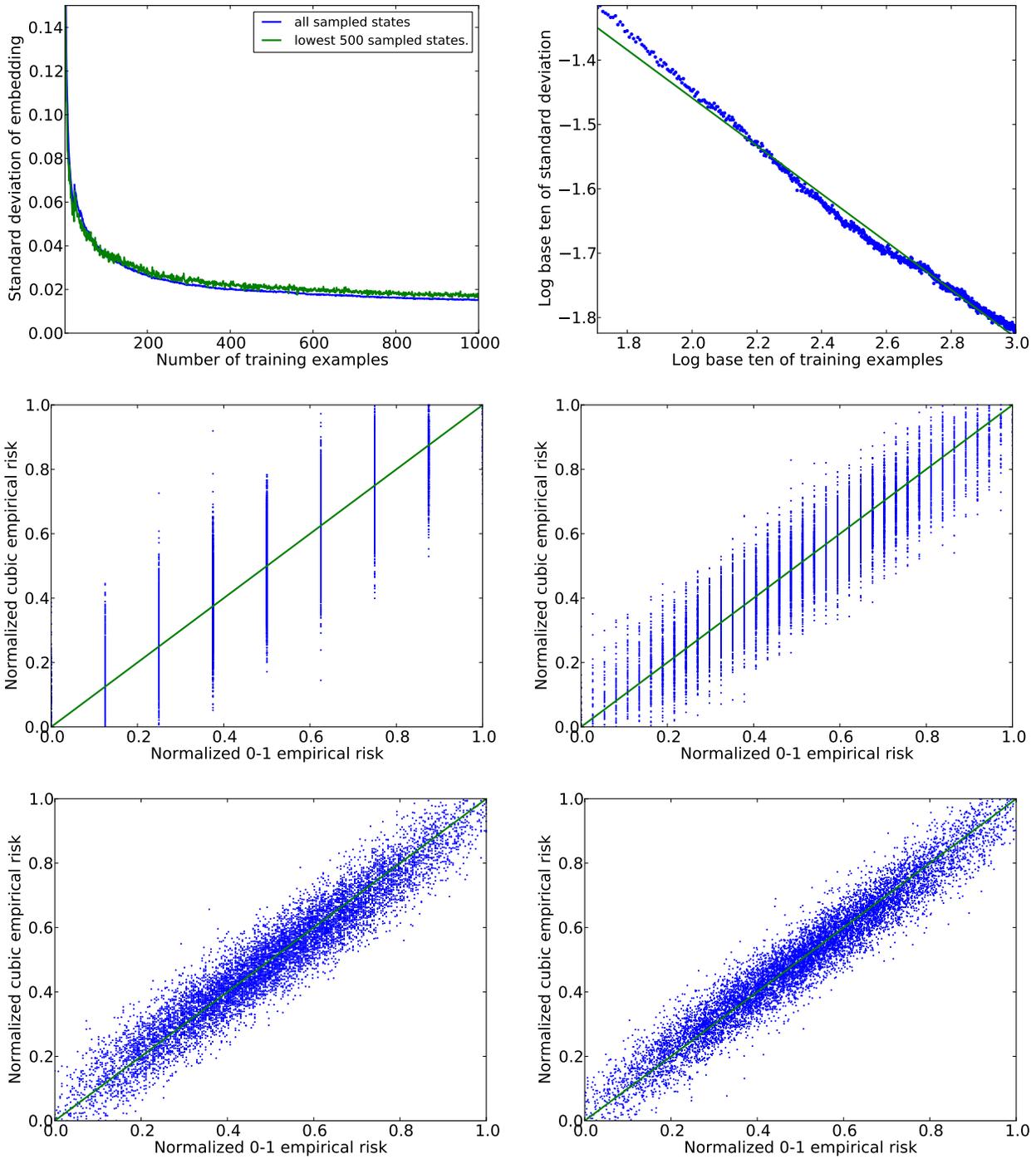


Figure 6: These plots quantify the convergence of the empirical risk embedding error as a function of the number of training examples. The upper-left plot is made by fitting the cubic loss function and evaluating the resultant embedding error on adult9 using a variable number of training examples. Here, the error is the standard deviation of the energy landscapes defined by the limited number of training examples. At each point, 10^4 states are sampled at random to evaluate the error. We also show the error in only the lowest 50 of these states to give an indication of the rate at which the low energy subspace is converging. On the upper-right, is a log plot of the same data indicating that embedding appears to converge as roughly $O(m^{-1/3})$. The remaining plots show correlations between the total empirical risk of the 10^4 randomly selected states using 0-1 loss and the total empirical risk on those states using cubic loss. These four plots were obtained by fitting the cubic loss function to the adult9 data set using: 10 training examples, 100 training examples, 1,000 training examples and 10,000 training examples. The error in these embeddings are 0.107, 0.0818, 0.062 and 0.055, respectively.

6.4. Estimated qubit requirements

Table 1: Upper-bounds on qubit requirements for selected problems

Loss function degree	#Features	Bit-depth	#Qubits
cubic	100	1	2,550
cubic	100	4	40,200
cubic	500	1	62,750
cubic	500	4	1,001,000
cubic	2,500	1	1,563,750
cubic	2,500	4	25,005,000

6.5. Data summary

Table 2: Data summary

Name	Dims	#Examples	Density (%)	Baseline error (%)
Long-Servedio	21	2000	100.00	50.00
Mease-Wyner	20	2000	100.00	49.80
covertypes	54	581012	22.20	36.46
mushrooms	112	8124	18.75	48.20
adult9	123	48842	11.30	23.93
web8	300	59245	4.20	2.92

6.6. Hyperparameters

Table 3: values of λ and ω offered to cross-validation

λ & ω
2.000000
0.398965
0.079583
0.015875
0.003167
0.000632
0.000126
0.000025
0.000005
0.000001

Table 4: ω values for sixth-order loss picked by cross-validation

Data set name	Label noise (%)				
	0	10	20	30	40
Long-Servedio	0.000001	0.000001	0.000001	0.000001	0.000632
Mease-Wyner	0.398965	0.000126	0.000126	0.000025	0.000005
covertypes	2.000000	2.000000	0.398965	2.000000	0.000025
mushrooms	2.000000	2.000000	2.000000	0.000632	0.000005
adult9	0.000001	2.000000	0.000025	0.079583	2.000000
web8	0.000632	0.000126	0.398965	0.398965	2.000000

Table 5: q values for q -loss picked by cross-validation

Data set name	Label noise (%)				
	0	10	20	30	40
Long-Servedio	0	-0.39	-0.24	-0.71	-0.55
Mease-Wyner	0	-2.96	-1.62	-1.36	0
covertime	-0.63	-0.54	-0.38	-0.5	-0.51
mushrooms	0	-0.76	-0.47	-0.17	-0.13
adult9	-0.86	-0.53	-0.43	-0.53	-0.07
web8	-0.99	-0.46	-0.41	-0.19	0

Table 6: C values for liblinear picked by cross-validation

Data set name	Label noise (%)				
	0	10	20	30	40
Long-Servedio	0.499978	2.506486	0.499978	0.499978	0.499978
Mease-Wyner	40000.00	0.499978	315.7562	12.565498	62.99213
covertime	0.499978	2.506486	62.99213	1000000.0	12.56541
mushrooms	2.506486	12.56541	0.499978	0.499978	0.499978
adult9	0.499978	62.992126	0.499978	0.499978	0.499978
web8	315.7562	0.499978	12.56541	12.56541	0.499978

Table 7: λ values picked by cross-validation for 0% label noise

Data set name	Method				
	logistic	square	smooth hinge	q-loss	sixth-order
Long-Servedio	0.003167	0.079583	0.015875	0.015875	0.000001
Mease-Wyner	0.000001	0.000025	0.000001	0.000126	2.000000
covertime	0.000025	0.000025	0.000001	0.000025	0.000632
mushrooms	0.000001	0.000025	0.000632	0.000025	0.398965
adult9	0.000001	0.000632	0.000126	0.003167	0.015875
web8	0.000001	0.000005	0.000001	0.000632	0.015875

Table 8: λ values picked by cross-validation for 10% label noise

Data set name	Method				
	logistic	square	smooth hinge	q-loss	sixth-order
Long-Servedio	0.000005	2.000000	0.003167	0.015875	0.000001
Mease-Wyner	0.000005	0.000632	0.000005	0.000126	0.398965
covertime	0.000025	0.000632	0.000126	0.000001	0.000001
mushrooms	0.000005	0.000001	0.000005	0.003167	0.398965
adult9	0.000632	0.003167	0.000126	0.015875	0.000632
web8	0.000005	0.000126	0.000005	0.000632	0.015875

Table 9: λ values picked by cross-validation for 20% label noise

Data set name	Method				
	logistic	square	smooth hinge	q-loss	sixth-order
Long-Servedio	2.000000	2.000000	2.000000	0.000126	0.000001
Mease-Wyner	0.000025	0.000005	0.000126	0.000126	0.079583
covertime	0.000001	0.000126	0.000126	0.000001	0.000025
mushrooms	0.000126	0.000632	0.000025	0.003167	0.398965
adult9	0.079583	0.079583	0.003167	0.015875	2.000000
web8	0.000001	0.000001	0.000126	0.000632	0.015875

Table 10: λ values picked by cross-validation for 30% label noise

Data set name	Method				
	logistic	square	smooth hinge	q-loss	sixth-order
Long-Servedio	2.000000	2.000000	2.000000	0.003167	0.000001
Mease-Wyner	0.000005	0.000001	0.000005	0.000126	0.000632
covertime	0.000001	0.000126	0.000025	0.000025	0.398965
mushrooms	0.000632	0.003167	0.000632	0.003167	0.079583
adult9	2.000000	0.003167	2.000000	0.003167	0.000632
web8	0.000126	0.000001	0.000126	0.000632	0.000005

Table 11: λ values picked by cross-validation for 40% label noise

Data set name	Method				
	logistic	square	smooth hinge	q-loss	sixth-order
Long-Servedio	2.000000	2.000000	2.000000	0.003167	0.000632
Mease-Wyner	0.000001	0.000005	0.000025	0.000126	0.000632
covertime	0.000001	0.000001	0.000001	0.000001	0.079583
mushrooms	0.000126	0.000632	0.003167	0.003167	0.003167
adult9	0.000126	0.000126	0.079583	0.000025	0.000025
web8	0.015875	0.079583	0.000632	0.000632	0.000001

References

- Babbush, Ryan, O’Gorman, Bryan, and Aspuru-Guzik, Alán. Resource efficient gadgets for compiling adiabatic quantum optimization problems. *Annalen der Physik*, 525(10-11):877–888, 2013.
- Barahona, Francisco. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.
- Barends, R, Kelly, J, Megrant, A, Veitia, A, Sank, D, Jeffrey, E, White, TC, Mutus, J, Fowler, AG, Campbell, Chen, Y, Chen, Z, Chiaro, B, Dunsworth, A, Neill, C, O’Malley, P, Roushan, P, Vainsencher, A, Wenner, J, Korotkov, AN, Cleland, AN, and Martinis, J. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, 2014.
- Bishop, Christopher M. Pattern recognition and machine learning (information science and statistics). 2006, 2006.

- Boixo, Sergio, Rønnow, Troels F, Isakov, Sergei V, Wang, Zhihui, Wecker, David, Lidar, Daniel A, Martinis, John M, and Troyer, Matthias. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218–224, 2014.
- Boros, Endre and Gruber, Aritanan. On quadratization of pseudo-boolean functions. In *ISAIM*, 2012.
- Brooke, J, Bitko, D, Rosenbaum, T, and Aeppli, G. Quantum annealing of a disordered magnet. *Science (New York, NY)*, 284(5415):779, 1999.
- Denchev, Vasil, Ding, Nan, Neven, Hartmut, and Vishwanathan, SVN. Robust classification with adiabatic quantum optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 863–870, 2012.
- Farhi, Edward, Goldstone, Jeffrey, Gutmann, Sam, and Sipser, Michael. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.
- Feldman, Vitaly, Guruswami, Venkatesan, Raghavendra, Prasad, and Wu, Yi. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012.
- Johnson, MW, Amin, MHS, Gildert, S, Lanting, T, Hamze, F, Dickson, N, Harris, R, Berkley, AJ, Johansson, J, Bunyk, P, Chapple, EM, Enderud, C, Hilton, JP, Karimi, K, Ladizinsky, E, Ladizinsky, N, Oh, T, Perminov, I, Rich, C Thom, MC, Tolkacheva, E Truncik, CJS, Uchaikin, S, Wang, J, Wilson, B, and Rose, B. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.
- Kadowaki, Tadashi and Nishimori, Hidetoshi. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.
- Kirkpatrick, Scott, Vecchi, MP, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- Kitaev, Alexei Yu, Shen, Alexander, and Vyalıy, Mikhail N. *Classical and quantum computation*. Number 47. American Mathematical Soc., 2002.
- Long, Philip M and Servedio, Rocco A. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010.
- Manwani, Naresh and Sastry, PS. Noise tolerance under risk minimization. *Cybernetics, IEEE Transactions on*, 43(3):1146–1151, 2013.
- Mease, David and Wyner, Abraham. Evidence contrary to the statistical view of boosting. *The Journal of Machine Learning Research*, 9:131–156, 2008.
- Rennie, Jason. On l2-norm regularization and the gaussian prior. 2003.
- Santoro, Giuseppe E, Martoňák, Roman, Tosatti, Erio, and Car, Roberto. Theory of quantum annealing of an ising spin glass. *Science*, 295(5564):2427–2430, 2002.
- Shor, Peter W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.
- Somma, RD, Boixo, S, Barnum, H, and Knill, E. Quantum simulations of classical annealing processes. *Physical review letters*, 101(13):130504, 2008.
- Somma, Rolando D, Nagaj, Daniel, and Kieferová, Mária. Quantum speedup by quantum annealing. *Physical review letters*, 109(5):050501, 2012.