Success Probabilities of $L_2$-norm Regularized Babai Detectors and Maximization

Xiao-Wen Chang, Qincheng Lu and Yingzi Xu
School of Computer Science, McGill University
3480 University Street, Montreal, H3A 0E9, Canada
Email: chang@cs.mcgill.ca, {qincheng.lu, yingzi.xu}@mail.mcgill.ca

Abstract—This paper is concerned with success probabilities of some $L_2$-norm regularized Babai detectors for MIMO detection. First, we extend the often used MMSE-SIC detector to two $L_2$-norm regularized Babai detectors. Then, we derive the corresponding success probability formulae of the two detectors. To increase their success probabilities, we propose to maximize them with respect to the regularization parameter vector by a machine learning based method called ML-LA, as conventional optimization methods are very inefficient. This moves the exponential computational cost to offline training, and the method outputs an approximation to the optimal regularization parameter vector in polynomial time. Finally, we present some numerical simulations to demonstrate the performance of the new detectors.

I. INTRODUCTION

Suppose that we have the linear model:

$$y = Ax^* + v,$$

where $y \in \mathbb{R}^m$ is an observation vector, $A \in \mathbb{R}^{m \times n}$ is a deterministic model matrix, the vector $x^*$ is an unknown integer parameter vector, whose elements are i.i.d. uniformly distributed over the finite integer set

$$X := \{ \pm 1, \pm 3, \ldots, \pm (M - 1) \}, \quad M = 2^k,$$

and $v \in \mathbb{R}^m$ is a noise vector distributed from $\mathcal{N}(0, \sigma_v^2I_m)$ for some given $\sigma_v > 0$. Our goal is to detect $x^*$. This is the real version of the detection problem in MIMO applications.

There are different methods to detect $x^*$. A review of commonly used detection techniques can be found in [1]. One method is to solve the integer version of the minimum mean squares error (MMSE) problem:

$$\min_{x \in X^n} \| y - Ax \|^2_2 + \lambda^* \| x \|^2_2, \quad (3)$$

where $\lambda^* = \sigma_v / \sigma_x$ with $\sigma_x^2 = (M^2 - 1) / 3$ being the variance of any entry of $x^*$. Note that (3) is an $L_2$-regularized integer least squares problem and we can rewrite it as

$$\min_{x \in X^n} \| y - [A \lambda^* I_n]_+ x \|_2^2, \quad (4)$$

Since it may be too time consuming to solve (4), which is NP-hard, one often finds a sub-optimal solution with low complexity. An often used sub-optimal solution is the MMSE-SIC detector (see, e.g., [2, Sec. 4.1.3]), to be denoted by $x^{(\lambda^*)}$.

In the following we show how to compute it. Suppose we have the compact QR factorization of the augmented matrix:

$$\begin{bmatrix} A \\ \lambda^* I_n \end{bmatrix} = QR, \quad (5)$$

where $Q \in \mathbb{R}^{(m+n) \times n}$ satisfies $Q^T Q = I_n$ and $R \in \mathbb{R}^{n \times n}$ is upper triangular with positive diagonal entries. Then, with

$$\tilde{y} := Q^T \begin{bmatrix} y \\ 0 \end{bmatrix}, \quad (6)$$

it is easy to show that (4) is equivalent to

$$\min_{x \in X^n} \| \tilde{y} - Rx \|_2^2.$$
success probability. To deal with the theoretical shortcoming, we transform the model to a new one so that the noise vector is uncorrelated. Then we define the corresponding Babai point denoted by $\mathbf{x}^{(A)}$ and give an exact formula for its success probability. We consider maximizing $\Pr(\mathbf{x}^{(A)} = \mathbf{x}^*)$ and $\Pr(\mathbf{x}^{(A)} = \mathbf{x}^*)$ with respect to $\lambda$. Because using a conventional optimization method to solve the optimization problems may be too time consuming, we propose a machine learning based method called ML-$\lambda$ to solve the problems. Numerical results will be given to demonstrate the effectiveness of the method and the performance of the two detectors.

II. SUCCESS PROBABILITY OF SOME BABAI POINTS

Note that the model (1) is equivalent to
\[
\begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} A \\ 0 \end{bmatrix} \mathbf{x}^* + \begin{bmatrix} \mathbf{v} \\ -\Lambda \mathbf{x}^* \end{bmatrix},
\]
where $\Lambda = \text{diag}(\lambda)$. Here we assume that the coefficient matrix of $\mathbf{x}^*$ has full column rank – it is certainly true if $\lambda > 0$ no matter whether $\Lambda$ has full column rank or not. It is also true when $\Lambda$ has full column rank even if $\lambda = 0$.

Then we compute the QR factorization:
\[
\begin{bmatrix} A \\ 0 \end{bmatrix} = QR,
\]
where $Q$ and $R$ have the same properties as those in (5).

Define
\[
\tilde{y} := Q^T \begin{bmatrix} y \\ 0 \end{bmatrix}, \quad \tilde{v} := Q^T \begin{bmatrix} \mathbf{v} \\ -\Lambda \mathbf{x}^* \end{bmatrix}.
\]

Then (8) is transformed to
\[
\tilde{y} = R \mathbf{x}^* + \tilde{v}.
\]

Based on $\tilde{y}$ and $R$ above, we can calculate $\mathbf{x}^{(A)}$ as (7).

To find the success probability of $\mathbf{x}^{(A)}$, first we present the distribution of $\tilde{v}$ in (11) in the following lemma. We omit the proof due to space limitation.

**Lemma 1.** Partition the Q-factor $Q \in \mathbb{R}^{(m+n)\times n}$ of the QR factorization (9) into two sub-matrices:
\[
Q = \begin{bmatrix} \tilde{Q} \\ \tilde{Q} \end{bmatrix}, \quad \tilde{Q} \in \mathbb{R}^{m \times n}, \quad \tilde{Q} \in \mathbb{R}^{n \times n}.
\]

Then $\tilde{Q} = AR^{-1}$ and $\tilde{Q} = \Lambda R^{-1}$. If $\mathbf{x}^*$ is deterministic, then
\[
\tilde{v} \sim \mathcal{N}(-\tilde{Q}^T \mathbf{x}^*, \sigma_v^2 \tilde{Q}^T \tilde{Q}).
\]

In particular, for $i = 1, \ldots, n$,
\[
\tilde{v}_i \sim \mathcal{N}(-\tilde{Q}_i^T \mathbf{x}^*, \sigma_v^2 \|\tilde{Q}_i\|_2^2),
\]
where $\tilde{Q}_i$ and $\tilde{Q}_i$ are the i-th columns of $\tilde{Q}$ and $\tilde{Q}$, respectively.

Note that if $\lambda = 0$ and $\Lambda$ has full column rank, then $\tilde{Q} = 0$ and $\tilde{Q}^T \tilde{Q} = I_n$. Thus, in this case, from (13) $\tilde{v} \sim \mathcal{N}(0, \sigma_v^2 I_n)$.

**Theorem 1.** If $\mathbf{x}^*$ is deterministic and $\mathbf{x}^* \in \mathcal{X}^n$, then, with the same notation as used in Lemma 1, the success probability of $\mathbf{x}^{(A)}$ satisfies
\[
\Pr(\mathbf{x}^{(A)} = \mathbf{x}^*) \approx \prod_{i=1}^n \rho_i(\mathbf{x}^*),
\]
where
\[
\rho_i(\mathbf{x}^*) = \frac{1}{2} \left( 1 + \text{erf} \left( \frac{r_{ii} - (\tilde{Q}_i^T \mathbf{x}^* + \tilde{v}_i)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) \right),
\]
otherwise
\[
\Pr(\mathbf{x}^{(A)} = \mathbf{x}^*) = \frac{1}{2} \left( \text{erf} \left( \frac{r_{ii} + (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) + \text{erf} \left( \frac{r_{ii} - (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) \right)
\]
with the error function $\text{erf}(\zeta) := \frac{2}{\sqrt{\pi}} \int_0^\zeta \exp(-t^2)dt$.

**Proof:** By the multiplication rule of probability,
\[
\Pr(\mathbf{x}^{(A)} = \mathbf{x}^*) = \prod_{i=1}^n \Pr(x_i^{(A)} = x_i^* \mid E_{i+1}),
\]
where the event $E_{i+1} = (x_j^* = x_j^*, j = i+1, \ldots, n)$.

From (11) we have
\[
\tilde{y}_i = r_{ii}x_i^* + \sum_{j=i+1}^n r_{ij}x_j^* + \tilde{v}_i, \quad i = 1, \ldots, n.
\]

Substituting $\tilde{v}_i$ in (7) (where the error function (17) is changed to (Λ)) by the right hand side of (17), we obtain
\[
c_i = x_i^* + \sum_{j=i+1}^n r_{ij}(x_j^* - x_j^*)/r_{ii} + \tilde{v}_i/r_{ii}.
\]

Then, when $E_{i+1}$ occurs, we obtain by Lemma 1 that
\[
e_i - x_i^* = \tilde{v}_i/r_{ii} \sim \mathcal{N}(E \{\tilde{v}_i\}/r_{ii}, \text{var}[\tilde{v}_i]/r_{ii}^2) = \mathcal{N}(-\tilde{Q}_i^T \mathbf{x}^*/r_{ii}, \sigma_v^2 \|\tilde{Q}_i\|_2^2/r_{ii}^2).
\]

Here the distribution of $\tilde{v}_i/r_{ii}$ given in (18) is actually an approximation because it ignores the correlation between $\tilde{v}_i$ and $\tilde{v}_j$ for $j = i+1, \ldots, n$, although it is assumed that $E_{i+1}$ has occurred. When $x_i^* = -(M - 1)$,
\[
\Pr(x_i^{(A)} = x_i^* \mid E_{i+1}) = \Pr(c_i - x_i^* \in (-\infty, 1] \mid E_{i+1}) \approx \frac{1}{2} \left( 1 + \text{erf} \left( \frac{r_{ii} + (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) \right).
\]

When $x_i^* \in \{\pm 1, \pm 3, \ldots, \pm (M - 3)\}$,
\[
\Pr(x_i^{(A)} = x_i^* \mid E_{i+1}) = \Pr(c_i - x_i^* \in (-1, 1] \mid E_{i+1}) \approx \frac{1}{2} \left( \text{erf} \left( \frac{r_{ii} + (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) + \text{erf} \left( \frac{r_{ii} - (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) \right).
\]

When $x_i^* = M - 1$,
\[
\Pr(x_i^{(A)} = x_i^* \mid E_{i+1}) = \Pr(c_i - x_i^* \in (-1, \infty) \mid E_{i+1}) \approx \frac{1}{2} \left( 1 + \text{erf} \left( \frac{r_{ii} - (\tilde{Q}_i^T \mathbf{x}^*)}{\sqrt{2} \sigma_v \|\tilde{Q}_i\|_2} \right) \right).
\]

Therefore, (15) follows from (16).
To overcome the correlation problem with $\tilde{\nu}$ in (11), we are going to transform (11) to a new linear model. For simplicity, we assume that $A$ in (1) has full column rank. Then from Lemma 1 we observe that $\tilde{Q}$ has full column rank. We compute its QL factorization: $Q = QL$, where $Q^TQ = I_n$ and $L \in \mathbb{R}^{n \times n}$ is lower triangular with positive diagonal entries. Then from (11) we obtain

$$y = Rx^* + \nu,$$

$$\bar{y} := L^{-T}y, \quad \bar{R} := L^{-T}R, \quad \nu := L^{-T}\nu,$$  

where $\bar{y}$ follows the normal distribution with $E\{\bar{y}\} = L^{-T}Q^TAx^*$, $\text{cov}\{\nu\} = L^{-T}\sigma^2_{\nu}Q^TQL^{-1} = \sigma^2_{\nu}L$. From the new model (19) we can define the corresponding new Babai point $x^{(A)}$. Following the proof for Theorem 1, we can show the following result.

**Theorem 2.** Suppose in (1) $x^* \in \mathcal{X}^n$ is deterministic and $A$ has full column rank. Then, with $B := QL^{-1}$ the success probability of $x^{(A)}$ corresponding to the model (11) satisfies

$$Pr(x^{(A)} = x^*) = \prod_{i=1}^{n} \tilde{\rho}_i(x^*),$$  

where

$$\tilde{\rho}_i(x^*) = \left\{ \begin{array}{ll} \frac{1}{2} \left(1 + \text{erf} \left( \frac{r_{ii} - (b_i^TAx^*)/\sigma_{\nu}}{\sqrt{2}\sigma_{\nu}} \right) \right), & x^*_i = \pm(M-1), \\ \frac{1}{2} \left( \text{erf} \left( \frac{r_{ii} + (b_i^TAx^*)/\sigma_{\nu}}{\sqrt{2}\sigma_{\nu}} \right) + \text{erf} \left( \frac{r_{ii} - (b_i^TAx^*)/\sqrt{2}\sigma_{\nu}}{\sqrt{2}\sigma_{\nu}} \right) \right), & \text{otherwise} \end{array} \right.$$  

with $b_i$ being the $i$-th column of $B$.

**Theorem 3.** Suppose in (1) $x^*$ is uniformly distributed over $\mathcal{X}^n$ and $A$ has full column rank and. Then, with the same notation used in Theorems 1 and 2, we have

$$Pr(x^{(A)} = x^*) \approx f(\lambda; A, \sigma_{\nu}) := \frac{1}{M^n} \sum_{x \in A^n} \prod_{i=1}^{n} \rho_i(x),$$  

$$Pr(x^{(A)} = x^*) = \tilde{f}(\lambda; A, \sigma_{\nu}) := \frac{1}{M^n} \sum_{x \in A^n} \prod_{i=1}^{n} \tilde{\rho}_i(x).$$  

**Proof:** From the law of total probability we have:

$$Pr(x^{(A)} = x^*) = \sum_{x \in A^n} Pr(x^{(A)} = x) \cdot Pr(x^* = x)$$

$$= \frac{1}{M^n} \sum_{x \in A^n} Pr(x^{(A)} = x)$$

which, with Theorem 1, leads to (22). Similarly we can show that (23) holds using Theorem 2.

If we take $\lambda = 0$ in Theorem 3, we can obtain the following corollary.

**Corollary 1.** If $x^*$ is uniformly distributed over $\mathcal{X}^n$ and $A$ has full column rank, then the success probability of the unregularized Babai point $x^{(0)}$ has the following formula:

$$Pr(x^{(0)} = x^*) = \frac{1}{M^n} \prod_{i=1}^{n} \left[1 + (M-1)\text{erf} \left( \frac{r_{ii}}{\sqrt{2}\sigma_{\nu}} \right) \right].$$  

**Remark 1.** In communications, typically error rates are used to measure the quality of a detector. The success probability of a detector is equal to 1 minus the vector error rate.

### III. ML-$\lambda$: A Machine Learning Based Method to Maximizing Success Probabilities

In this section we consider maximizing $f(\lambda; A, \sigma_{\nu})$ and $\tilde{f}(\lambda; A, \sigma_{\nu})$ (see Theorem 3) with respect to $\lambda$. Because the two functions are similar, we consider only $f(\lambda; A, \sigma_{\nu})$. The method to be proposed can be applied to maximize $f(\lambda; A, \sigma_{\nu})$ as well.

From (22), we see that the cost of computing $f(\lambda; A, \sigma_{\nu})$ grows exponentially with $n$. When $n$ is large, numerically finding the optimal $\lambda$ by maximizing $f(\lambda; A, \sigma_{\nu})$ directly is too time consuming, as a conventional optimization algorithm requires computing $f(\lambda; A, \sigma_{\nu})$ probably many times. However, machine learning based methods can be developed to shift the main computational cost of a task to an offline training phase, running in lower time complexity during its application. Recently, machine-learning communities have shown the effectiveness of amortized optimization through learning to optimize, which gives a predictor that efficiently infers solutions for repeated optimization problems with similar structures all at once [7], [8]. In particular, methods that learn a continuous hidden space of solutions based on a neural network architecture called variational autoencoder (VAE) [9] have been successfully applied to various fields [10]–[12]. We adopt the basic idea of VAE to estimate the optimal $\lambda$ in this paper. For different optimization problems, the VAE methods are different. We will design a specific VAE method, to be called ML-$\lambda$, to solve our optimization problem. Essentially our learning-based approach transforms the difficult optimization problem to a much easier one. It will produce a good approximation to the optimal $\lambda$ upon providing $A, \sigma_{\nu}$ and $\lambda^*$ at a very low cost.

**A. Optimization as a learning problem**

The goal of our learning-based method is to learn a meaningful mapping between $\lambda$ and representation $z \in \mathbb{R}^h$ for given $(A, \sigma_{\nu})$ in a problem-specific latent space, and to learn parameters in the predictor. In this data-driven approach, a training dataset $T_{\text{train}} = \{(\lambda_i, f(\lambda_i; A^{(i)}, \sigma_{\nu}^{(i)}), A^{(i)}, \sigma_{\nu}^{(i)})\}$, is generated for finding parameters in the neural network. This offline data simulation phase often takes a long time, because for each randomly sampled $(A^{(i)}, \sigma_{\nu}^{(i)})$, we find the optimal $\lambda_i$ to maximize $f(\lambda; A^{(i)}, \sigma_{\nu}^{(i)})$ by a conventional optimization algorithm. However, the application phase has a time complexity of $O(hm^2 + h^2n)$.

In applying the conditional VAE (CVAE) (see [13], [14]), we introduce a hidden variable $z$ with a prior $p(z)$ for $\lambda$, given $A$ and $\sigma_{\nu}$. Assuming we have defined a likelihood function $p_{\theta}(\hat{A}|z, A, \sigma_{\nu})$ with a parameter vector $\theta$ (see later), we intend to find the posterior $p_{\theta}(z|\lambda, A, \sigma_{\nu})$ for the inference purpose. One attempt to compute it is to use $p_{\theta}(z|\lambda, A, \sigma_{\nu}) = p_{\theta}(A|z, A, \sigma_{\nu})p(z)/p_{\theta}(\lambda|A, \sigma_{\nu})$ and $p_{\theta}(\lambda|A, \sigma_{\nu}) = \int p(z)p_{\theta}(\lambda|z, A, \sigma_{\nu})dz$. However, this
is computational intractable due to the integration. Instead, CVAE finds its approximation \( q_\phi(z|\lambda, A, \sigma_v) \) parameterized by vector \( \phi \) (see later). To find \( \theta \) and \( \phi \), we will use the so-called evidence lower bound (ELBO) \([9]\) as a part of our training loss (see (26) and (27)):

\[
\text{ELBO}(\lambda, A, \sigma_v)_{\theta, \phi} = E_{z \sim q_\phi(z|\lambda, A, \sigma_v)} \{ \log p_\theta(z|A, \sigma_v) \} - D_{KL}(q_\phi(z|\lambda, A, \sigma_v) || p(z)),
\]

where \( D_{KL}(\cdot) \) refers to the Kullback-Leibler divergence. It is known that the maximization of (25) is equivalent to the minimization of \( D_{KL}(\cdot) \).

The whole process is known as the so-called evidence lower bound (ELBO) \([9]\) as a part of our training process. An explanation of each component is given as follows.

1) Encoder: It first produces a distribution \( q_\phi(z|\lambda, A, \sigma_v) \) of the hidden representation \( z \) of \( \lambda \) given the instance \((A, \sigma_v)\), which approximates the true posterior \( p_\theta(z|A, \sigma_v) \). Then it outputs a sample \( z \) from \( q_\phi \) via the re-parameterization trick \([9]\). The whole process is known as encoding. We implement "Encoder" using convolutional neural networks (CNNs) \([15]\), multilayer perceptrons (MLPs) and an attention mechanism (ATTENTION) \([16]\).

2) Decoder: It converts an output \( z \) of "Encoder" for the input \((A, \lambda, \sigma_v)\) back to the \( \lambda \)-space. "Decoder" is trained to generate an output \( \lambda_{\text{ML}} \) that closely matches \( \lambda \). Now we define the log-likelihood function:

\[
\log p_\theta(z|A, \sigma_v) := -\|\lambda - \lambda_{\text{ML}}\|^2_2,
\]

which is used in (25).

3) Predictor: It produces \( \alpha_\psi(z, A, \sigma_v) \) as an approximation to \( f(\lambda, A, \sigma_v) \) to avoid evaluating the latter at the inference time. The parameter vector \( \psi \) is found in the training phase (see later).

In order to let "Encoder", "Decoder" and "Predictor" embed the problem instance in the same way, we force "Problem Embedding" (PE) component in "Encoder", "Decoder" and "Predictor" to share the same parameters, which is part of \( \phi \). Parameters in "ATTENTION" in the three parts are also identical to reduce the number of total parameters to train.

B. Training

The purpose of training is to optimize the parameters \( \{\phi, \theta, \psi\} \) by minimizing the loss function over the training set \( T_{\text{train}} \). We define the loss for the \( i \)-th training sample as:

\[
\ell_i(\phi, \theta, \psi) := \beta(\alpha_\psi(z^{(i)}, A^{(i)}, \sigma_v^{(i)}) - f(\hat{\lambda}^{(i)}; A^{(i)}, \sigma_v^{(i)}))^2
- (1 - \beta)\text{ELBO}(\hat{\lambda}^{(i)}, A^{(i)}, \sigma_v^{(i)})_{\theta, \phi},
\]

where \( z^{(i)} \) is the output of Encoder with \( \hat{\lambda}^{(i)}, A^{(i)}, \sigma_v^{(i)} \) as the input, and \( \beta \in (0,1) \) is a hyper-parameter adjusting the contribution from the prediction loss and the negative ELBO.

Based on (26), the loss function for training is defined as:

\[
\mathcal{L}(\phi, \theta, \psi) := \frac{1}{|T_{\text{train}}|} \sum_{i \in T_{\text{train}}} \ell_i(\phi, \theta, \psi),
\]

where \( T_{\text{train}} \) is the training set of samples.

C. Inference

At this stage, \( \{\phi, \theta, \psi\} \) are fixed. Given \((A, \sigma_v)\), we first find the hidden representation \( z^* \) of \( \lambda^* \) using the trained "Encoder". Then we employ the gradient-ascent method to maximize \( \alpha_\psi(z, A, \sigma_v) \) with respect to \( z \) as the initial point. The gradient \( \partial \alpha_\psi(z, A, \sigma_v)/\partial z \) can be calculated efficiently using automatic differentiation and its complexity can be bounded by \( O(b^2n) \) \([17]\). The number of iterations can be set as a small constant. Let the solution be denoted by \( z_{\text{ML}} \). We expect the output \( \lambda_{\text{ML}} \) of "Decoder" for the input \( (z_{\text{ML}}, A, \sigma_v) \) will outperform \( \lambda^* \).

D. Experiments

In our experiments, we set \( k = 2 \) in (2) and use (22) and (23) to compute the theoretical success probabilities of the two detectors \( x^{(\lambda)} \) and \( x^{(\hat{\lambda})} \). We generate two types of matrices, which are the real versions of the channel matrices in communications.

1) \( A = \text{randn}(m, n) \). We take \( \sigma_v = [\text{randn}] \). If \( P_r(x^{(\lambda)} = x^*) - P_r(x^{(\lambda')} = x^*) \geq 0.04 \), the corresponding instance is taken as a testing sample.

2) \( A = \Psi^{1/2}H\Phi^{1/2} \), where \( H = \text{randn}(m, n), \psi_{ij} = a_{i,j} \) and \( \phi_{ij} = b_{i,j} \) with \( a, b \in [0, 1], \Psi^{1/2} = \Phi^{1/2} = \Phi \); see \([18]\) and \([19]\). In our tests, we choose \( a = 0.95, b = 0.97 \) to make these matrices significantly more ill-conditioned than \( A \) in Type 1. We take \( \sigma_v = [\text{randn}] \ast 0.08 \). If \( P_r(x^{(\lambda)} = x^*) - P_r(x^{(\lambda')} = x^*) \geq 0.02 \), the corresponding instance is taken as a testing sample.
A imposed success probability gap. \( \lambda \) the performance of testing sample selection criterion ensures that we can observe \( f(x) \) entries are uniformly and independently sampled from \( \lambda \). When \( x \) success probability is the ratio of the number of occurrences This gives (2). For each \( X \) instances of \( x \sim N(\bar{\lambda}, \sigma_v^2) \). Thus, there are a total of four

In our experiments, \( \lambda = 8 \times 10^{4} \) to \( \min \) can bring. Our

Our test network function. The training is completed within 2 hours on a NVIDIA GeForce RTX 3080 GPU.

The test results for the success probabilities of the two Babai detectors corresponding to different \( \lambda \) are given in Fig. 2. In the figure, the prefixes “Exp” and “The” represent “experimental” and “theoretical” success probabilities, respectively. In each plot, the 100 samples with each corresponding to one pair \( (A, \sigma_v) \) are ordered according to the experimental success probability of \( x^{(\lambda_{\text{ML}})} \). Thus the same data samples in plots (a) and (b) are in different order, and so are the same data samples in plots (c) and (d).

From Fig. 2 we observe that (1) Although the formula (22) is an approximation, the experimental and theoretical success probabilities of \( x^{(\lambda_{\text{ML}})} \) are close. (2) \( x^{(\lambda^*)} \) performs (much) better than \( x^{(0)} \) and our extended detector \( x^{(\lambda^*)} \) performs (much) better than \( x^{(\lambda_{\text{ML}})} \), especially for Type 1 matrices. These indicate that the conventional \( L_2 \)-norm regularization is helpful and our extension can bring significant improvement. (3) In general, the success probability of \( x^{(\lambda_{\text{ML}})} \) obtained by our \( L_2 \)-\( \lambda \)-method is a reasonable approximation to the success probability of \( x^{(\lambda^*)} \) and is higher than the success probability of \( x^{(\lambda)} \). (4) There is no significant difference between the performance of \( x^{(\lambda^*)} \) and \( x^{(\lambda)} \) and comments (2) and (3) above can be applied to \( x^{(\lambda_{\text{ML}})} \) as well.

IV. SUMMARY AND FUTURE WORK

We have extended the MMSE-SIC detector to two \( L_2 \)-regularized Babai detectors by replacing the regularization constant \( \lambda^* \) with an \( n \)-dimensional vector \( \lambda \) and presented two success probability formulae for the two detectors. Optimizing the success probabilities with respect to \( \lambda \) by a conventional optimization method would have exponential time complexity. To overcome this issue, we have proposed machine learning based method \( ML-\lambda \), which has polynomial time complexity, to estimate the optimal \( \lambda \). In the future, we plan to replace the model matrix \( A \) with the \( R \)-factor of its QR factorization in the training input so that the learning based detector trained for a fixed \( n \) can be applied to various values of \( m \).
REFERENCES


