Coded Machine Learning: Joint Informed Replication and Learning for Linear Regression

Shahroze Kabir, Frederic Sala, Guy Van den Broeck, and Lara Dolecek
{shkabir, fredsala}@ucla.edu, guyvdb@cs.ucla.edu, dolecek@ee.ucla.edu
UCLA, Los Angeles, CA 90095

***Abstract*—This paper is concerned with coded machine learning: protecting machine learning algorithms from noise in test data by an informed channel coding approach. Unlike with traditional data storage, we do not seek to ensure that all test data is correctly read from storage and used as a noiseless input to the algorithm. Rather, we seek to protect data in a way that minimizes the effect on the algorithm output (i.e., minimizes a loss compared to the hypothetical noiseless output). We focus on the case where the collected test data, derived from low-power sensors and devices, is inherently noisy. We show that a smart replication strategy is an effective choice to reduce the impact on the algorithm output for linear regression algorithms. We focus on two scenarios. The first case is where the regression model is fixed, and we must allocate a fixed budget of redundancy for our replication scheme (in order to minimize the loss on the output due to noisy test data). Analyzing this case is necessary to build our understanding for the second case which is more novel. The second case involves a scenario where we may learn an optimized model and jointly protect it. We illustrate the advantages of our approach with practical experiments.**

***Index Terms*—Channel Coding, Machine Learning, Repetition Coding, Linear Regression.**

I. Introduction

The recent surge in the popularity of machine learning is a testament to the power and flexibility of the techniques the field offers. A particularly exciting application of machine learning relates to low-power devices, energy-efficient sensors, wireless networks, etc. For example, such devices are involved in localization and object tracking, security and intrusion detection, media access control, and fault detection.

Low-power devices collect data and transmit it to a central processor (sometimes called the fusion center) where the machine learning models are trained and implemented. Due to the low-power, cost efficient nature of the devices, the data collection process along with the transmission link tend to be noisy. As a result, the algorithm performed at the central processor operates on noisy data. The purpose of this paper is to detail how to tackle this noise.

The traditional approach to dealing with noise involves applying error-correcting codes. In our scenario, however, we face several concerns:

- Low-power and low-cost devices may not admit a powerful (energy-consuming) error-correction architecture,
- The measurements available at the sensor may already be noisy (e.g., due to measurement noise, limited resolution sampling, etc.) and thus the noiseless data is not available for encoding, and
- Classically, the goal is to protect all of the data equally well, but a machine learning model may be influenced by some parts of the data far more than others, so that traditional coding methods are not efficient.

We briefly discuss each of these issues and the role they play in the solution we propose in this paper. We begin with the last concern. The idea here is that error-correction typically seeks to protect all of the data. The application of this data is not considered in the error protection strategy; in fact, the error-correction strategy typically lies in an entirely different abstraction layer. Such an approach is sufficient for general-purpose devices, but in our case, it is not efficient.

Consider, for example, a model with tens of thousands of binary features. A small number of these features may have a very large impact on the model output; such features require significant protection from even a small amount of noise. On the other hand, the vast majority of features may only have a very small impact on the final output, so that these features are resilient to noise. Clearly, a different coding strategy should be implemented for the two sets of features.

Our goal in this work (building on our previous work in [1], [2], [3]) is to introduce a coding strategy that takes the machine learning algorithm into account. We assume that the fusion center has a high-quality model that has access to noiseless training data; however, this model will operate on noisy test data transmitted by the remote low-power measurement devices[1]. The noise in the test data is assumed to be Gaussian in nature as the noise introduced at the output of these low power sensors is often modeled as Gaussian. We seek to develop schemes that tailor the error-protection

Research supported in part by NSF grants CAREER CCF-1150212, CCF-1527130, NSF (#IIS-1657613 and #IIS-1633857), and DARPA (#N66001-17-2-4032)

[1]More precisely, we consider two cases. In the first case, the high-quality model is fixed and we can only change the error protection for the noisy features. Analyzing this helps us to build up to the second, more novel scenario. In the second case, the model is jointly trained from the noiseless training data and the test data noise characteristics in order to minimize the expected loss when using noisy data. Here too, we further reduce the impact of noise by optimizing the error protection on the test features.

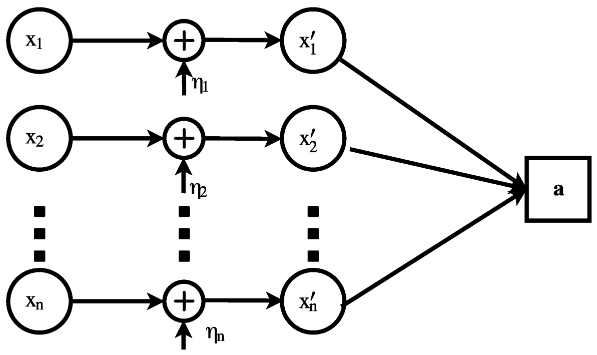

Fig. 1. Noise model for linear regression.

strategy used by these remote sensors in order to minimize the noise impact on the fusion center's model output.

What type of error-correcting codes suit our needs? Here we must recall the first and second issues described above. Firstly, low-cost devices with energy constraints imply that we must use simple, easily-decoded classes of codes. Secondly, the fact that the noiseless data is not available suggests the use of replication coding. With replication (repetition), the processor can simply ask for more samples as an encoding technique, never relying on the (unavailable) noiseless data. We will attempt to apply these principles to designing codes for protecting linear regression test data.

We use the following example to build intuition regarding the technique that can be used to improve the performance of the algorithm. Consider a small data set generated from an underlying system where the model is $x = 0.5y + 16z$, where y and z are the measured features which have noise added to them. When the features are corrupted equally by zero mean Gaussian noise with a variance of 4, then the mean squared loss at the output of the algorithm is 1025. With uniform repetition coding of 3 replications for each feature, the loss is reduced to 341.7. However by exhaustive search we can find that the optimal redundancy allocation is 1 units to y and 5 units to feature z. This yields a mean squared loss of at the output 205.8. This simple example motivates the need for a strategy to efficiently optimize coding (that is, redundancy allocations).

The rest of this paper is organized as follows. In Section II we briefly describe related work. In Section III we present our noise model and the use of replication. In Section IV we present our metric to measure the performance of the algorithm and how to optimize a redundancy allocation given regression coefficients using convex and submodular optimization techniques. Building on this, in Section V we present how to jointly optimize the linear regression model (that is, the regression coefficients) and redundancy allocation during the training phase using submodular optimization techniques. We present experimental results in Section VI and conclude in Section VII.

II. Related work

There is an abundance of papers studying robust machine learning. For example, the authors in [4] study the experimental performance of several algorithms with artificially noisy datasets. Machine learning algorithms also have to deal with either missing features or features that have been corrupted, the effects of which were studied in [5]. In [6], a game theoretic approach to avoid an over reliance on particular features was discussed. However these papers do not consider coding strategies for improving the performance of the algorithms. In contrast, we use coding theory strategies to protect the data in our application.

Research has also been done based on measuring the "correctness" of the output of the algorithm when the data is known to be noisy. The authors in [7] define the notion of "same decision probability" in order to measure the probability of making the same decision for Bayesian networks when additional information is known. In [8] the authors study classification techniques when the labels are known to be noisy. Our own work measure the importance of features in terms of how much redundancy needs to be allocated to them.

On the other hand, there is a rich body of work regarding protection of data from different types of noise using coding techniques. This involves coding for data storage mediums such as disk drives [9], flash memory devices [10], non volatile memories [11] and solid state drives [12]. There are also papers proposing coding to improve the robustness of wireless sensor networks [13], [14]. However, as mentioned the techniques in these papers do not take advantage of the extra application-specific information that may be known.

There has been a rising interest regarding developing application-specific coding techniques to solve particular problems. For example, in the problem posed by the authors in [15] for a distributed learning system, some nodes communicate to the central fusion center slower than others. The authors in [15] provide a method called Gradient Coding which involves transmitting a linear combination of data from each node to counteract the effects of some nodes lagging behind others. The authors in [16] study codes for distributed coding. They introduce techniques that are able to recover lost data when a storage node goes down in a distributed storage system. Our own work is in the same vein as these papers as we attempt to address the problem of noisy data for linear regression with coding theory strategies.

This paper builds on our previous work presented in [1], [2] and [3]. In [1], we used an approximation of the mean absolute error (MAE) as the metric to optimize via repetition coding. In [2] and [3], we presented an efficient way of computing the approximate mean absolute loss for linear classifiers and present submodular techniques for optimizing redundancy allocation. In contrast, in this paper, we presents techniques to optimize the actual mean squared loss for regression. We also present a technique to optimize the allocation of redundancy during training, a novel and previously not considered idea.

III. Noise model and regression codes

We begin by establishing notation for linear regression and some preliminaries regarding the noise models used in this paper. We define a feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ where we consider elements $x_j \in \mathbb{F}_2$ (for binary regression) and $x_j \in \mathbb{R}$ (for regression with continuous values). We define the vector of regression coefficients $\mathbf{a} = (a_1, \ldots, a_n)$ where $a_j \in \mathbb{R}$. The target value corresponding to the jth feature vector is given by $y_j \in \mathbb{R}$. We also define the matrix $\mathbf{X}$ of p feature vectors and the corresponding vector $\mathbf{y} \in \mathbb{R}^p$ containing the corresponding p target values. For a given feature vector $\mathbf{x}$, linear regression predicts the value of the target variable y as $\mathbf{a}^T\mathbf{x}$.

The noise model that we consider is shown in Figure 1. The noisy version of feature vector $\mathbf{x}$ is denoted by $\mathbf{x}'$. The learning algorithm receives noisy versions of each feature. A Gaussian random variable η_j with mean zero and variance σ_j^2 is added to the jth feature. We use Gaussian noise, as the output of low power sensors used in wireless sensor networks is often corrupted by Gaussian noise when it arrives at the fusion center. Depending on the quality of the sensor, different sensors may have different levels of noise. Therefore our model is able to handle different levels of noise for different features.

A. Replication

Next we discuss the effects of channel coding on the noise applied to each feature. As we previously described, we rely on replication for two reasons: first, to deal with the fact that the encoder may not have access to the noiseless, true data, and, secondly, because replication codes enable us to provide an appropriate level of protection to each feature. Replication involves repeating the jth feature k_j times, i.e., the codeword $c_j = (x_j, x_j, \ldots, x_j)$. The replicated values are passed through the noisy channel and are corrupted. Depending on the type of noise model, replication coding mitigates the probability of feature error in different ways. For our Gaussian noise model, the decoding process involves taking the average of the noisy versions of the repetitions of a feature: $\frac{1}{k_j}(\sum_{i=1}^{k_j} c_j(i))$. This effectively reduces the overall variance of the noise ${\sigma_j}^2$ of the jth feature:

$$Var\left[\frac{1}{k_j}\sum_{i=1}^{k_j}\left(c_j(i)+\eta_j(i)\right)\right] = \frac{1}{{k_j}^2}Var\left[\sum_{i=1}^{k_j}\eta_j(i)\right] = \frac{1}{{k_j}^2}(k_j{\sigma_j}^2) = \frac{{\sigma_j}^2}{k_j}.$$

Thus replication reduces the variance of the noise added to the feature by a factor equal to the number of repetitions.

Now that we have defined how replication reduces the effect of noise on the data, we discuss how to allocate redundancy given a budget of N replications in order to reduce the effects of noise on the predictions made by the algorithm in an informed way.

IV. Expected Noise Loss

We measure the expected loss due to noise (between the output of the noisy algorithm and the noiseless algorithm) with a mean-squared error (MSE) approach. The MSE is the following:

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{a}^T\mathbf{x} - \mathbf{a}^T\mathbf{x}'\right\|\right] \\
&= \mathbb{E}[(\mathbf{a}^T(\mathbf{x}-\mathbf{x}'))^T(\mathbf{a}^T(\mathbf{x}-\mathbf{x}'))] \\
&= \mathbb{E}[((\mathbf{x}-\mathbf{x}')^T\mathbf{a}\mathbf{a}^T(\mathbf{x}-\mathbf{x}'))] \\
&= Tr(\mathbf{a}\mathbf{a}^T Cov(\mathbf{x}-\mathbf{x}')) + \mathbb{E}[\mathbf{x}-\mathbf{x}']^T\mathbf{a}\mathbf{a}^T\mathbb{E}[\mathbf{x}-\mathbf{x}'].
\end{aligned} \tag{1}$$

The final step uses the identity that for any matrix $\mathbf{A}$ and vector $\mathbf{c}$ we have

$$\mathbb{E}[\mathbf{c}^T\mathbf{A}\mathbf{c}] = Tr(\mathbf{A}Cov(\mathbf{c})) + \mathbb{E}[\mathbf{c}]^T\mathbf{A}\mathbb{E}[\mathbf{c}].$$

We note that using the MSE (as opposed to MAE) is motivated in part by the fact that linear regression involves minimizing square error itself.

A. Optimizing the MSE loss

Next we analyze the continuous version of the optimization problem. Using the same idea as the expression in (1), we derive the expected loss for the current scenario. The combined noise vector η is a zero mean Gaussian random vector, with covariance matrix $\mathbf{\Sigma} = diag(\sigma_1^2, \ldots, \sigma_n^2)$. The expected loss is given by

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{a}^T\mathbf{x} - \mathbf{a}^T\mathbf{x}'\right\|\right] \\
&= Tr(\mathbf{a}\mathbf{a}^T Cov(\mathbf{x}-\mathbf{x}')) + \mathbb{E}[\mathbf{x}-\mathbf{x}']^T\mathbf{a}\mathbf{a}^T\mathbb{E}[\mathbf{x}-\mathbf{x}'] \\
&= \sum_{j=1}^{n} a_j^2{\sigma_j}^2.
\end{aligned}$$

We are now ready to introduce the key optimization for our informed replication approach. We have a budget of N repetitions and we wish to allocate these among features in such a way that our resulting expected loss is minimized. We define the redundancy allocation vector $\mathbf{k} = (k_1, ..., k_n)$, where the ith feature is replicated k_i times for $1 \leq i \leq n$. We set up the constrained optimization problem as follows:

$$\min_{k_1,\ldots,k_n} g(\mathbf{k}) = \sum_{j=1}^{n}\frac{a_j^2{\sigma_j}^2}{k_j},$$

subject to

$$\sum_{j=1}^{n} k_j = N.$$

where k_j is a positive integer.

Note that due to the integer constraint, the problem is not convex. However, if we relax the constraint and allow k_j to be real valued, the optimization problem becomes convex. We solve this optimization problem using the method of Lagrange multipliers.

Theorem 1 *The optimal redundancy allocation for the continuous relaxation of the problem for the jth feature is given by*

$$k_j = \frac{|a_j\sigma_j|}{\sum_{j=1}^{n}|a_j\sigma_j|}N.$$

Proof: Define the function g' and as follows:

$$g'(\mathbf{a},\mathbf{k},\lambda) = \sum_{j=1}^{n}\frac{a_j^2{\sigma_j}^2}{k_j} - \lambda(\sum_{j=1}^{n}k_j - N).$$

Taking the partial derivatives and setting them to zero gives the following relationship between λ and k_j

$$k_j = |a_j\sigma_j|\sqrt{\frac{-1}{\lambda}}.$$

Substituting this back into the sum constraint gives the following solution for λ:

$$\lambda = \frac{-(\sum_{j=1}^{n}|a_j\sigma_j|)^2}{N^2}.$$

We then solve for k_j by substituting the value of λ in the equation for k_j

$$\begin{aligned} k_j &= |a_j\sigma_j|\sqrt{\frac{-N^2}{-(\sum_{j=1}^{n}|a_j\sigma_j|)^2}} \\ &= \frac{|a_j\sigma_j|}{\sum_{j=1}^{n}|a_j\sigma_j|}N. \end{aligned}$$

■

Note that if this solution yields integer values for all k_j's, then this is the optimal solution for the integer version of the problem as well. However, if any of obtained values are not integers, then we do the following. Use $\lfloor k_j \rfloor$ for all values of j. There are at most $n-1$ units left after this for allocation. We can try all possible allocations for these bits to find the solution to the integer version of the problem.

B. Solution using submodular optimization

While the continuous relaxation gives an efficient technique to approach the optimal solution for our optimization problem, it still requires an exhaustive search to allocate up to $n-1$ of the final redundancy units. To solve this problem efficiently, we must take the integer constraints into consideration. As such, we use submodular optimization. Submodular optimization is a technique used to solve optimization problems that involve integer constraints [17]. It is a computationally efficient and nearly optimal technique, as shown in [17]. The algorithm, which is a greedy technique, involves allocating units of redundancy one at a time iteratively. Given m available units of redundancy, consider the redundancy allocation vector $\mathbf{k}(i) = (k_1(i),\ldots,k_n(i)), 1 \le i \le m$ which denotes the units of redundancy allocated to each vector at the ith step of the algorithm. Then at step $i+1$, we assign an additional unit of redundancy to the lth feature. This gives the vector $\mathbf{k}(i+1) = (k_1(i),\ldots,k_l(i)+1,\ldots,k_n(i))$. The goal of the minimization at each step is to find the $\min_l \sum_{j=1}^{n}\frac{a_j^2{\sigma_j}^2}{k_j}, 1 \le l \le n$. This process continues until all units of redundancy are allocated.

We show that our optimization function is submodular. The objective function must fulfill the conditions of monotonicity and diminishing returns in order to validate the use of submodular optimization. The monotonicity statement is as follows:

Lemma 1 *Given redundancy allocation vectors* $\mathbf{k} = (k_1,\ldots,k_i,\ldots,k_n)$ *and* $\tilde{\mathbf{k}} = (k_1,\ldots,k_{i-1},\tilde{k}_i,k_{i+1},\ldots,k_n)$, *where* $k_i > \tilde{k}_i$. *Then* $f(\mathbf{k}) < f(\tilde{\mathbf{k}})$.

The proof for Lemma 1 is immediate, as the optimization function is inversely proportional to k_i for all i. Lemma 1 shows that the cost function decreases as more units of redundancy are allocated. Next we consider the diminishing returns property. For this property to hold, repeated allocation of redundancy units to the same feature must yield reduced improvements.

Lemma 2 *Consider redundancy allocation vectors* $\mathbf{k} = (k_1,\ldots,k_n)$ *and* $\mathbf{z} = (z_1,\ldots,z_n)$, *where* $k_j \le z_j$ *for all* j. *Then for all* i ,

$$\begin{aligned} &\sum_j a_j^2\sigma_j^2/k_j - (\sum_{j\neq i} a_j^2\sigma_j^2/k_j + a_i^2\sigma_i^2/(k_i+1)) \ge \\ &\quad \sum_j a_j^2\sigma_j^2/z_j - (\sum_{j\neq i} a_j^2\sigma_j^2/z_j + a_i^2\sigma_i^2/(z_i+1)). \end{aligned}$$

Proof: We prove the lemma by expanding the summation on both sides and canceling out terms accordingly. This simplifies the expression to

$$\begin{aligned} &a_i^2\sigma_i^2/k_i - a_i^2\sigma_i^2/(k_i+1) \ge a_i^2\sigma_i^2/z_i - a_i^2\sigma_i^2/(z_i+1) \\ &\Rightarrow 1/k_i - 1/(k_i+1) \ge 1/z_i - 1/(z_i+1)). \end{aligned}$$

This final statement holds true when $k_i \le z_i$ and k_i, z_i are integers, which is the indeed the case. ■

Thus the cost function is indeed submodular. We use the floor of the redundancy allocations obtained from the continuous relaxation and then allocate the remaining $n-1$ units using submodular optimization.

V. Optimizing the linear regression model

So far we have worked on optimizing the redundancy allocation given a fixed set of regression coefficients. This reflects the assumption that we have a fixed model, without the ability to change it. On the other hand, as long as we know the test data noise characteristics, we can *optimize* the model in order to minimize the joint square error (from regression and noise simultaneously).

Motivated by the previous notion, in this section we show how we optimize the model for regression given the regression data and the noise model. Afterwards, we further select the optimal replication scheme. We assume that the noise on each feature is independent of noise on all other features. Let $\mathbf{X} \in \mathbb{R}^{p\times n}$ be the matrix of p data points with n features. Let the matrix $\mathbf{X}' = \mathbf{X} + \mathbf{N}$ be the noisy data set where $\mathbf{N} \in \mathbb{R}^{p \times n}$ is a matrix where each row is an independent realization of an n-variate Gaussian random vector with zero mean and $\mathbf{\Sigma}$ as the covariance matrix. Assume that each element of this random vector is independent and as such, the covariance matrix is a diagonal matrix, i.e. $\mathbf{\Sigma} = diag(\sigma_1^2, \ldots, \sigma_n^2)$. Then $E[\mathbf{N}] = \mathbf{0}$ and $E[\mathbf{N}^T\mathbf{N}] = p\mathbf{\Sigma}$ (by the expectation of the Wishart distribution [18]). The objective function to minimize in this case is as follows:

$$\begin{aligned}
&\min_a \mathbb{E}[\|\mathbf{y} - \mathbf{X}'\mathbf{a}\|^2] \\
&\quad = \min_{\mathbf{a}} \mathbb{E}[(\mathbf{y} - (\mathbf{X}+\mathbf{N})\mathbf{a})^T(\mathbf{y} - (\mathbf{X}+\mathbf{N})\mathbf{a})] \\
&\quad = \min_{\mathbf{a}} \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbb{E}[(\mathbf{X}+\mathbf{N})]\mathbf{a} \\
&\quad - \mathbf{a}^T\mathbb{E}[(\mathbf{X}+\mathbf{N})^T]\mathbf{y} + \mathbf{a}^T\mathbb{E}[(\mathbf{X}+\mathbf{N})^T(\mathbf{X}+\mathbf{N})]\mathbf{a} \\
&\quad = \min_{\mathbf{a}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2 + p\mathbf{a}^T\Sigma\mathbf{a}.
\end{aligned}$$

The problem takes the form of the regularized least squares problem. The solution to the above regression problem is as follows:

$$\mathbf{a} = (\mathbf{X}^T\mathbf{X} + p\mathbf{\Sigma})^{-1}\mathbf{X}^T\mathbf{y}.$$

Now we plug this value of $\mathbf{a}$ back into the objective function and attempt to optimize over the elements of the matrix $\mathbf{\Sigma}$. While we have reduced the terms that involve components of $\mathbf{\Sigma}$ for the optimization, we must still approximate the inverse of $(\mathbf{X}^T\mathbf{X} + p\mathbf{\Sigma})$ if we wish to find a solution to the optimization problem for the noise variance. Plugging in the value for the optimal $\mathbf{a}$ and simplifying gives the following cost function to minimize:

$$\begin{aligned}
&\min_{\sigma_1,\ldots,\sigma_n} \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + p\mathbf{\Sigma})^{-1}\mathbf{X}^T\mathbf{y} \\
&\qquad = \min_{\sigma_1,\ldots,\sigma_n} \mathbf{y}^T(\mathbf{I} - \mathbf{X}(\mathbf{X}^T\mathbf{X} + p\mathbf{\Sigma})^{-1}\mathbf{X}^T)\mathbf{y}.
\end{aligned}$$

Note that the objective function expression cannot be negative as it is derived from the linear regression cost, which is a non-negative value. Thus, the solution to the above optimization problem is also the solution of the following problem:

$$\max_{\sigma_1,\ldots,\sigma_n} \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + p\mathbf{\Sigma})^{-1}\mathbf{X}^T\mathbf{y}.$$

A. Solution by submodular optimization

Once again, we attempt to solve the optimization problem using submodular optimization. Before we begin we state several facts that will be used to prove that the submodular optimization approach is applicable.

Fact 1 *[19] Let the matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Then for all $\mathbf{x} \in \mathbb{R}^n$ $\lambda_n\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T\mathbf{A}\mathbf{x} \leq \lambda_1\mathbf{x}^T\mathbf{x}$.*

This fact allows us to bounds the quadratic form involving a matrix with its eigenvalues. We next look at the bounds on the eigenvalues for a matrix that has been perturbed.

Fact 2 *[20] Let $\mathbf{A}, \mathbf{A} + \mathbf{E} \in \mathbb{R}^{n \times n}$ be symmetric matrices and $\lambda_k(\mathbf{G})$ denote the kth largest eigenvalue of the matrix $\mathbf{G}$. Then $\lambda_n(\mathbf{E}) + \lambda_k(\mathbf{A}) \leq \lambda_k(\mathbf{A} + \mathbf{E}) \leq \lambda_1(\mathbf{E}) + \lambda_k(\mathbf{A})$.*

Finally we also state a fact that relates the eigenvalues of a matrix with its inverse.

Fact 3 *[19] Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a symmetric non-singular matrix. Let $\lambda_1 \geq \ldots \geq \lambda_n$ be the eigenvalues of $\mathbf{A}$. Then the eigenvalues of $\mathbf{A}^{-1}$ are $\frac{1}{\lambda_n} \geq \ldots \geq \frac{1}{\lambda_1}$.*

Now we are ready to prove the submodularity properties for the joint optimization. We begin by stating the lemma required to show that successive allocations of redundancy units causes the objective function to decrease.

Lemma 3 *Given two redundancy allocations defined by the covariance matrices $\mathbf{\Sigma} = p \times diag(\frac{\sigma_1^2}{k_1}, \ldots, \frac{\sigma_i^2}{k_i}, \ldots, \frac{\sigma_n^2}{k_n})$ and $\tilde{\mathbf{\Sigma}} = p \times diag(\frac{\sigma_1^2}{k_1}, \ldots, \frac{\sigma_n^2}{k_{i-1}}, \frac{\sigma_i^2}{\tilde{k}_i}, \frac{\sigma_n^2}{k_{i+1}}, \ldots, \frac{\sigma_n^2}{k_n})$ respectively, where $\tilde{k}_i < k_i$ for a particular i where $1 \leq i \leq n$. We define $h(\mathbf{\Sigma}) = \mathbf{y}^T(\mathbf{I} - \mathbf{X}(\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma})^{-1}\mathbf{X}^T)\mathbf{y}$. Then $h(\mathbf{\Sigma}) \leq h(\tilde{\mathbf{\Sigma}})$.*

Proof: Firstly $\mathbf{y}^T\mathbf{y}$ is common to both $h(\mathbf{\Sigma})$ and $h(\tilde{\mathbf{\Sigma}})$. Next, we use the above defined facts to put bounds on the eigenvalues for inverse of a perturbed matrix.

$$\begin{aligned}
&\frac{1}{\lambda_{n-l+1}(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma})} \leq \lambda_l((\mathbf{X}^T\mathbf{X} - \mathbf{\Sigma})^{-1}) \\
&\leq \frac{1}{\lambda_{n-l+1}(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma})}.
\end{aligned}$$

Note that the indices are reversed due to the fact that we are defining the bounds of the eigenvalue of a matrix using the eigenvalues of its inverse. We define $\mathbf{u} = \mathbf{X}^T\mathbf{y}$. Then $\mathbf{u}^T(\mathbf{X}^T\mathbf{X} - \mathbf{\Sigma})^{-1}\mathbf{u} \leq \frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma})}\mathbf{u}^T\mathbf{u}$. Similarly, for $\tilde{\mathbf{\Sigma}}$ we have $\mathbf{u}^T(\mathbf{X}^T\mathbf{X} - \tilde{\mathbf{\Sigma}})^{-1}\mathbf{u} \leq \frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\tilde{\mathbf{\Sigma}})}\mathbf{u}^T\mathbf{u}$. As the covariance matrices are diagonal, we have the fact that $\lambda_n(\mathbf{\Sigma}) \leq \lambda_n(\tilde{\mathbf{\Sigma}})$. Therefore the quadratic form involving $\mathbf{\Sigma}$ is greater than or equal to the quadratic form involving $\tilde{\mathbf{\Sigma}}$. As such $h(\mathbf{\Sigma}) \leq h(\tilde{\mathbf{\Sigma}})$. ■

Next we wish to prove the diminishing returns property for this optimization problem. The diminishing returns property is as follows:

Lemma 4 *Given redundancy allocation vectors $\mathbf{k} = (k_1, \ldots, k_n)$ and $\mathbf{z} = (z_1, \ldots, z_n)$, where $k_j \leq z_j$ for all j where $1 \leq j \leq n$. These define two covariance matrices $\mathbf{\Sigma}_k = p \times diag(\sigma_1^2/k_1, \ldots, \sigma_n^2/k_n)$ and $\mathbf{\Sigma}_z = p \times diag(\sigma_1^2/z_1, \ldots, \sigma_n^2/z_n)$. We further define $\mathbf{\Sigma}_{k+1} = p \times diag(\sigma_1^2/k_1, \ldots, \sigma_i^2/(k_i+1), \ldots, \sigma_n^2/k_n)$ and $\mathbf{\Sigma}_{z+1} = p \times diag(\sigma_1^2/z_1, \ldots, \sigma_i^2/(z_i+1), \ldots, \sigma_n^2/z_n)$. Then for all i $h(\mathbf{\Sigma_k}) - h(\mathbf{\Sigma_{k+1}}) \geq h(\mathbf{\Sigma_z}) - h(\mathbf{\Sigma_{z+1}})$, where $1 \leq i \leq n$.*

Proof: Using the bounds established for the monotonicity proof, we attempt to show the diminishing returns property. Firstly we note that $\lambda_n(\mathbf{\Sigma_k}) - \lambda_n(\mathbf{\Sigma_{k+1}}) \leq \sigma_i^2/(k_i) - \sigma_i^2/(k_i+1)$. Also as $k_j \leq z_j$ for all j (where $1 \leq j \leq n$) and are integers, we have for all i (where $1 \leq i \leq n$):

$$\sigma_i^2/(k_i) - \sigma_i^2/(k_i+1) \geq \sigma_i^2/(z_i) - \sigma_i^2/(z_i+1).$$

Next we expand the following expression

$$h(\mathbf{\Sigma_k}) - h(\mathbf{\Sigma_{k+1}})$$
$$= \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma_{k+1}})^{-1}\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma_k})^{-1}\mathbf{X}^T\mathbf{y}.$$

We can use the upper bound for the first term and the lower bound for the second term to get

$$h(\mathbf{\Sigma_k}) - h(\mathbf{\Sigma_{k+1}})$$
$$\leq \left(\frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{k+1}})} - \frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_k})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}.$$

Similarly for $\mathbf{\Sigma_z}$ we have

$$h(\mathbf{\Sigma_z}) - h(\mathbf{\Sigma_{z+1}})$$
$$\leq \left(\frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{z+1}})} - \frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_z})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}.$$

We note that:

$$\left(\frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{k+1}})} - \frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{z+1}})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}$$

$$\geq \left(\frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_k})} - \frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_z})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}.$$

The above inequality is true due to the fact that the terms on the right hand side involve inverses of larger terms than that on the left hand side. We can rearrange the inequality to bring it into the desired form. In addition it is possible to show that the upper bounds on both sides of the inequality will be equally tight due to the fact that both sides represent costs within one additional unit of each other. Therefore we have:

$$\left(\frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{k+1}})} - \frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_k})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}$$

$$\geq \left(\frac{1}{\lambda_n(\mathbf{X}^T\mathbf{X}) + \lambda_n(\mathbf{\Sigma_{z+1}})} - \frac{1}{\lambda_1(\mathbf{X}^T\mathbf{X}) + \lambda_1(\mathbf{\Sigma_z})}\right)\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y},$$
$$\Rightarrow h(\mathbf{\Sigma_k}) - h(\mathbf{\Sigma_{k+1}}) \geq h(\mathbf{\Sigma_z}) - h(\mathbf{\Sigma_{z+1}}).$$

This concludes the proof for submodularity for the joint optimization. ■

Thus, given a budget of N repetition, we can use submodular optimization to efficiently allocate the redundancy optimally. We can now revisit the original example in Section I and apply this technique. Let us assume that the data is generated using the model ($x = 0.5y + 16z$). However, now the redundancy allocation must be done during the training phase. Given the fact that the noise variance vector is know to be $[1, 20]$ (very high noise on feature 2), with the same redundancy budget of 6 units, we find that the optimal redundancy allocation remains the same. However, the regression weights that are calculated change to 0.53 for feature y and 15.9 for feature z. We will see more examples of changes in regression weights in the experimental results section.

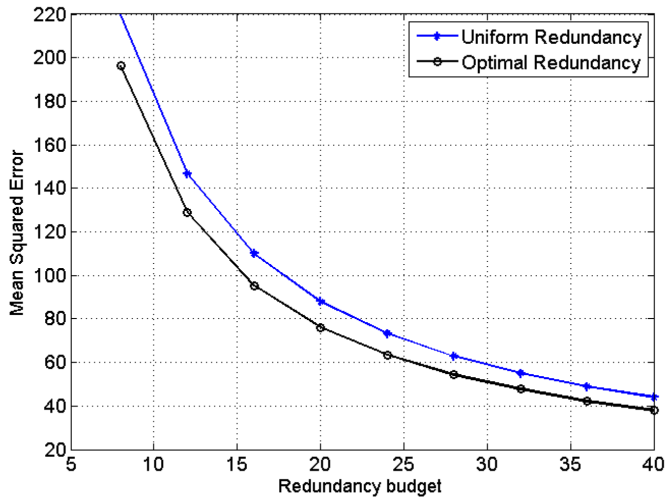

Fig. 2. MSE for Optimal Allocation vs Uniform Allocation for the 1st set of regression coefficients.

VI. Experimental Results

In this section we present experimental results which demonstrate the advantage our optimization methods provide (over an uninformed coding scheme). We first present results for allocating redundancy when the regression coefficients are fixed. We use the following regression model: $u = 3v + 9x + 7y + 4z$. The vector of variances is $[2, 2, 4, 4]$. For a redundancy budget of 40 units, the optimal redundancy allocation is $k_v = 4, k_x = 13, k_y = 15, k_z = 8$ units. The mean squared error for the optimal allocation is 38.03 and for uniform allocation, it is 44.

Next we look at how the cost varies as redundancy allocation increases from 8 units upwards. We plot the mean squared cost for the optimal allocation and the uniform allocation for the above setup. The plot is shown in Figure 2. We can see that the optimal allocation provides a clear benefit over the uniform allocation in terms of the MSE. Furthermore we see that benefit is present even at high values of redundancy. However there is the factor of diminishing returns which indicates that for a high enough redundancy budget, the optimal allocation would be the uniform allocation.

To further see the benefits of our allocation strategy, we use the following example: Let the regression model be $u = v + 9x + 7y + 4z$, with the noise variance vector as $[10, 10, 1, 2]$. This means that the noise in features 1 and 2 is very high, and the noise in features 3 and 4 is comparatively lower. For this case, the plot for the optimal MSE and uniform MSE over a range of values of the redundancy is shown in Figure 3. The advantage of our allocation scheme is very clear in this scenario. The optimal allocation scheme produces a MSE which is roughly half of the MSE for the uniform allocation for almost all values of the redundancy budget. As

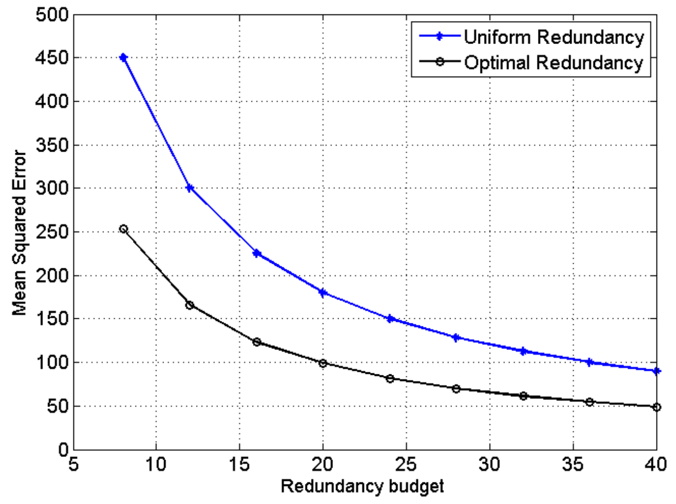

Fig. 3. MSE for Optimal Allocation vs Uniform Allocation for the 2nd set of regression coefficients.

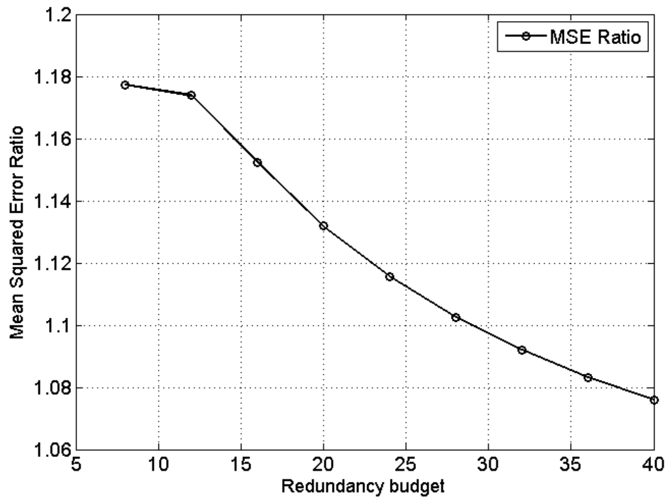

Fig. 4. Ratio of uniform MSE to optimal MSE for the power data set.

our allocation scheme takes into account both the regression weights and noise variance for each feature, it is able to provide a consistent benefit over the uniform scheme in this case. These two examples also illustrate how the amount of improvement in the model's performance depends on the noise distribution and model parameters.

Next we consider the case where the optimal allocation must be made during the training phase. We use the power production dataset [21]. The dataset contains 9568 data points collected from a power plant working at full load. It contains four features which are: hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V). The target variable is the net hourly output of the power plant. To start with, we consider the following noise variance vector $[8, 8, 1, 1]$. This means that the first two features are affected by noise with a variance of 8 units, while the remaining features are affected by noise with variance 1. We compute the optimal allocation of redundancy units when only 8 units of redundancy are available. The resulting redundancy allocation is $k_T = 5, k_{AP} = 1, k_{RH} = 1$, and $k_V = 1$. We can see that despite the fact that the first and second features are corrupted by noise with high variance, only the first feature gets the majority of the redundancy budget. The output MSE for this allocation is 27, while the output MSE for uniform allocation is 32.2, thus the optimal allocation provides about 18% improvement over the uniform case. We find that the regression weight vector (ignoring the intercept) $\mathbf{a}$ obtained using the optimal allocation is $[-1.7761, -0.2888, 0.1338, -0.1179]$. On the other hand, the noiseless training gives us the following vector of regression coefficients: $[-1.9689, -0.2316, 0.0772, -0.1620]$, showing that it is necessary to change the regression coefficients when the data is expected to be noisy.

Next we plot the ratio of uniform allocation MSE to optimal allocation MSE for a range of redundancy allocations for this setup. The plot is shown in Figure 4. We can see in the figure that using the submodular optimization technique gives a significant advantage over the uniform allocation for low values of redundancy budget. This advantage decreases as the budget increases, however it does not completely vanish. In scenarios where highly accurate predictions are required, even the slightest increase in accuracy is valuable. Our redundancy allocation technique ensures that the model is as accurate as possible with the given redundancy budget.

VII. Conclusion

In this paper, we considered informed replication techniques to protect linear regression test data from noise. We presented a strategy to compute the optimal redundancy allocation (in terms of the divergence from the hypothetical noise-free case on the output) for a regression model when applied to data that has been corrupted by Gaussian noise. We showed how this allocation of redundancy is a function of the regression coefficients and noise variance for each feature when the regression coefficients are given. We further presented a technique to obtain the optimal redundancy allocation for noisy features during the training stage. We showed how the regression coefficients become a function of the redundancy allocation while the model is being trained, along with the improvement of our redundancy allocation techniques compared to a uniform allocation. The optimal redundancy allocation is a step towards identifying levels of protection for each feature. Future work can involve application of these techniques to different noise models such as the binary symmetric channel and using this concept of protection to build more complex codes for regression.

References

[1] K. Mazooji, F. Sala, G. Van den Broeck, and L. Dolecek, "Robust channel coding strategies for machine learning data," in *Proc. of 54nd Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, Sep. 2016.

[2] F. Sala, S. Kabir, G. Van den Broeck, and L. Dolecek, "Don't Fear the Bit Flips: Optimized Coding Strategies for Binary Classification," arXiv preprint arXiv:1703.02641 (2017).

[3] F. Sala, S. Kabir, G. Van den Broeck, and L. Dolecek, "Dont Fear the Bit Flips: Robust Linear Prediction Through Informed Channel Coding," in *Reliable machine learning in the wild (WILDML 2017)*, August 2017.

[4] E. Kalapanidas, N. Avouris, M. Craciun, and D. Neagu, “Machine learning algorithms: a study on noise sensitivity,” in *Proc. 1st Balkan Conference in Informatics*, Thessaloniki, Greece, Nov. 2003.

[5] O. Dekel, and O. Shamir, “Learning to classify with missing and corrupted features, ” in *Proc. 25th Int. Conf. on Machine Learning*, Helsinki, Finland Jul. 2008.

[6] A. Globerson, and S. Roweis, “Nightmare at test time: robust learning by feature deletion,” in *Proc. 23rd Int. Conf. on Machine Learning*, Pittsburgh, PA, Jun. 2006.

[7] A. Choi, Y. Xue, and A. Darwiche, “Same-decision probability: A confidence measure for threshold-based decisions,” *Int. J. of Approximate Reasoning* vol. 53, no. 9, pp. 1415-1428, Dec. 2012.

[8] B. Frenay, and A. Kaban, “A comprehensive introduction to label noise,” in *Proc. European Symp. on Artificial Neural Networks, Comp. Intelligence and Machine Learning, ESANN*, Bruges, Belgium, Apr. 2014.

[9] C. M. Riggle, and S. G. McCarthy, “Design of error correction systems for disk drives,” *IEEE Trans. Magn*, vol 34, no. 4, pp. 2362-2371, Jul. 1998.

[10] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, “Codes for asymmetric limited-magnitude errors with application to multilevel flash memories,” *IEEE Trans. Inf. Theory*, vol 56, no. 4, pp. 1582-95, Apr. 2010.

[11] L. Dolecek, and F. Sala, “Channel coding methods for non-volatile memories,” *Foundations and Trends in Communications and Information Theory*, vol. 13, no. 1, pp. 1-28, Feb 2016.

[12] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng, “LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives,” in *Proc of Conf. on File and Storage Technologies (FAST 13)*, San Jose, CA Feb. 2013.

[13] G. Balakrishnan, M. Yang, Y. Jiang, Y. Kim, “Performance analysis of error control codes for wireless sensor networks,” in *Proc. IEEE Int. Conf. on Inf Theory*, Las Vegas, NV, Apr. 2007.

[14] S. Chung, and S. Lee, “Coding for Wireless Sensor Networks,” in *Smart Sensors for Health and Environment Monitoring*, Springer. Netherlands, 2015, pp 307-323.

[15] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, “Gradient Coding: Avoiding Stragglers in Synchronous Gradient Descent,” *stat.*, vol. 1050, Mar. 2017.

[16] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, “Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff,” in *IEEE Trans. Inf. Theory*, vol 58, no. 3, pp. 1837-52, Mar. 2012.

[17] A. Krause, and D. Golovin.(2014) *Submodular function maximization*[Online]. Available: http://www.cs.cmu.edu/afs/.cs.cmu.edu/Web/People/dgolovin/papers/submodular_survey12.pdf.

[18] J. Wishart, “The generalised product moment distribution in samples from a normal multivariate population,” *Biometrika*, pp. 32-52, Jul. 1928.

[19] A. Laub, *Matrix analysis for scientists and engineers*. SIAM, 2005.

[20] L. N. Trefethen, *Numerical linear algebra*. SIAM, 1997.

[21] P. Tfekci, “Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods,” *Int. J. of Elect. Power and Energy Syst.*, vol. 60, pp. 126-140, Sep. 2014.