Robust Channel Coding Strategies for Machine Learning Data

Kayvon Mazooji, Frederic Sala, Guy Van den Broeck, and Lara Dolecek
{kmazooji1, fredsala}@ucla.edu, guyvdb@cs.ucla.edu, dolecek@ee.ucla.edu
UCLA, Los Angeles, CA 90095

***Abstract*—Two important recent trends are the proliferation of learning algorithms along with the massive increase of data stored on unreliable storage mediums. These trends impact each other; noisy data can have an undesirable effect on the results provided by learning algorithms. Although traditional tools exist to improve the reliability of data storage devices, these tools operate at a different abstraction level and therefore ignore the data application, leading to an inefficient use of resources. In this paper we propose taking the operation of learning algorithms into account when deciding how to best protect data. Specifically, we examine several learning algorithms that operate on data that is stored on noisy mediums and protected by error-correcting codes with a limited budget of redundancy; we develop a principled way to allocate resources so that the harm on the output of the learning algorithm is minimized.**

***Index Terms*—Channel Coding, Machine Learning, Statistics, Optimization.**

I. Introduction

Machine learning is among the most important tools available for analyzing and predicting our world and the vast amount of data it produces. The study of statistical learning algorithms has experienced a renaissance over the last three decades. As these algorithms have become more and more popular, it has become common to execute them on inexpensive, imperfect hardware, while using data stored in noisy memories. These limitations can be viewed as departures from an idealized model of learning, where the input data is read from storage and transferred error-free and the hardware executing the algorithm is perfectly reliable.

The problem of dealing with noise in storage and computation has long been a target of study, dating back to the 1940's. Existing solutions can also be applied to learning; however, by considering machine learning problems and reliability questions separately, we may be incurring inefficiency. For this reason, we are interested in studying joint problems of learning and reliability. Specifically, in this work, we consider learning problems where some of the features are protected with redundant symbols (as in standard coding theory). We allow for only a finite budget of redundancy symbols; the key question is how to allocate this limited budget in order to minimize the effects of noise on the learning algorithm. This setup is depicted in Fig. 1. An intuitive idea is to assign a larger amount of redundancy to those features that are most important (and thus invest our reliability budget into the most critical information). However, it is not immediately clear how to do this; in this work we seek a principled approach.

Research supported in part by the NSF Graduate Research Fellowship Program and NSF grant CCF-1150212.

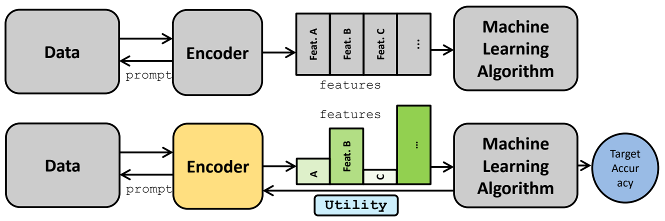

Fig. 1. The top half of the figure shows a block diagram for a conventional machine learning algorithm. The bottom half shows the modified version of the problem we consider in this work; the elements of the problem we modify are highlighted in color. Rather than applying a uniform reliability scheme (e.g., a single error-correcting code) to the feature data, we consider the impact of the reliability of each of these features on the algorithm itself.

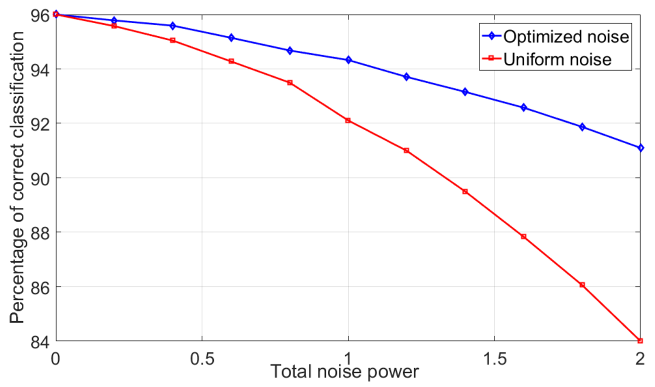

Fig. 2. Illustrative example for a naive Bayes classifier and Fisher's Iris data set [11]. Here, the features are corrupted by additive white Gaussian noise (AWGN) with total variance between 0 and 2. We optimize the allocation of this noise into features to illustrate the idea of non-uniform coding. The red curve shows the baseline approach, where the noise has equal strength among all features. The blue curve optimizes the noise allocation among features.

In Figure 2, we illustrate the potential power of our idea for a simple classification application, without assuming a particular coding method is used. In Figure 2, we show a naive Bayes classifier trained noiselessly on Fisher's classic Iris data set [11]. Afterwards, the features in the data set are affected by additive white Gaussian noise (AWGN) and re-classified as a test set. The total noise is fixed and can be distributed among the features in two ways: first, uniformly, so that each feature suffers equally distributed noise, and, secondly, in an optimized fashion, inversely proportional to a feature ranking based on an information gain ranking filter. The first technique illustrates a naive coding scheme where all features are equally protected, while the second demonstrates that different coding power should be allocated to different features. Observe that there is a significant impact on the performance of the classifier, particularly at higher noise levels.

A. Related work

There are a number of previous topics that interact with our problem setting. One related area is distributed learning. In [1], optimality guarantees are provided for distributed estimators in the cases where the nodes are isolated and where there is interactive communication between them. In [2], problems such as hypothesis testing and parameter estimation are considered for the multiterminal case where each terminal has data compression (rate) constraints. Unlike our work, these papers do not consider channel coding (or explicit coding schemes).

Another relevant area is feature selection. A general introduction to variable selection to help improve the performance, speed, or cost effectiveness of predictors is provided in [3]. Our work can be seen as a previously unexplored type of feature selection.

A number of previous works have considered the robustness or stability of various machine learning algorithms. For example, an experimental study of a variety of algorithms when data sets are artificially polluted with noise was performed in [4]. In [5], the case of missing, corrupted features was dealt with by introducing two machine learning techniques, based on a linear programming approximation and an online-to-batch strategy. Similarly, [6] avoided over-reliance on a particular feature (that could be missing at test time) in classification by using a game-theoretic approach. Here, classifiers were developed that are robust to feature deletion. Feature selection for submodular objective functions was considered in [7]. However, these works do not consider controlling robustness to noise through coding strategies.

Other papers have studied the value of information. For example, the robustness of decisions to hidden variables can be measured by computing the probability the same decision would have been made if these hidden variables were observed. This is the so-called "same-decision probability," introduced in [8]. The problem of label noise for classification algorithms is described in [9]. An interval-based approach to producing Bayes classifiers that are robust to missing data (without assumptions on the patterns of this data) was given by [10]. Our paper can be viewed as complementary to these efforts, as it also produces a value of information, but in the context of allocating protection against noise.

The rest of the paper is organized in the following way. In Section II, we introduce some preliminaries and notation. Afterwards, in Section III, we consider optimizing coding for linear regression algorithms. In Section IV, we comment on other applications and variants of the problem of coding for noisy data in learning algorithms. We conclude in Section V.

II. Preliminaries

In this section, we recall some basic facts about learning algorithms and introduce some simple coding ideas. We start with a *training set* with T data points $(\mathbf{x}^{(t)}, y^{(t)})$ where $\mathbf{x}^{(t)} \in \mathbb{R}^K$, $y^{(t)} \in \mathbb{R}$ and $K \in \mathbb{N}$, $t \in \{1, \ldots, T\}$. We call the $\mathbf{x}^{(t)}$'s *data points* and the $y^{(t)}$'s *target values*. Each entry in the vector $\mathbf{x}^{(t)}$ is referred to as a *feature*. The goal of regression algorithms is to predict the target value y for a new data point $\mathbf{x}$ based on the training set. This prediction follows from the choice of model; for example, linear regression models y as a linear function of a particular data point $\mathbf{x}$ such that $y = \mathbf{a}^\intercal\mathbf{x} + b$ for $\mathbf{a} \in \mathbb{R}^K$, $b \in \mathbb{R}$.

In classification problems, the same setting applies, but we force the $y^{(t)}$'s to be *class variables* with discrete values from the set $C = \{0, 1, \ldots, B-1\}$. In particular, the naive Bayes classifier makes the assumption that all the features are conditionally independent conditioned on the class variable. The maximum a-posteriori (MAP) estimate of the class of a point $\mathbf{x}$ is $\max_{c \in C} \prod_{i=1}^{K} p(x_i|c)$.

One of the most simple error-correcting codes is repetition coding. For a binary information vector $\mathbf{z}$ of length k, let $\mathbf{c}$ be the encoding of $\mathbf{z}$. We replace the bit z_i with $\mathbf{c}_i$, the bit z_i repeated multiple times. This code has the property that for each encoding $\mathbf{c}_i$, $\lceil |\mathbf{c}_i|/2 \rceil - 1$ bits are always correctable using maximum likelihood (ML) decoding (in this case, just taking the value of the majority of the bits). Thus, the entire codeword $\mathbf{c}$ allows for the correction of $\min_i \lceil |\mathbf{c}_i|/2 \rceil - 1$ errors.

Now, suppose that $\mathbf{c}$ is passed through a binary symmetric channel (BSC) with crossover probability ϵ_{BSC}. Then if $|\mathbf{c}_i|$ is odd, the probability of decoding error for the ith bit is

$$\sum_{j=\lceil |\mathbf{c}_i|/2 \rceil}^{|\mathbf{c}_i|} \binom{|\mathbf{c}_i|}{j} \epsilon_{\text{BSC}}^{j} (1-\epsilon_{\text{BSC}})^{|\mathbf{c}_i|-j}. \tag{1}$$

If $|\mathbf{c}_i|$ is even and $\frac{|\mathbf{c}_i|}{2}$ errors occur, we guess that z_i is 1 with probability .5. Regardless of the prior on z_i, we prove in 2 that the probability of error in this case is

$$\sum_{j=\lceil (|\mathbf{c}|_i - 1)/2 \rceil}^{|\mathbf{c}_i|-1} \binom{|\mathbf{c}_i|-1}{j} \epsilon_{\text{BSC}}^{j} (1-\epsilon_{\text{BSC}})^{|\mathbf{c}_i|-1-j}. \tag{2}$$

We thus conclude that if $|\mathbf{c_i}|$ is odd, adding another bit of redundancy does not decrease the probability of decoding error. However, if $|\mathbf{c_i}|$ is odd and we add two bits of redundancy, it is a well known fact that the probability of error strictly decreases. It follows that if $|\mathbf{c_i}|$ is even and we add two bits of redundancy, the probability of error strictly decreases.

III. Codes for Regression Data

In this section, we introduce coding strategies for the linear regression algorithm. The goal is to protect the noisy data points in a way that maintains the predictive power of the linear model. We may also wish to control the overall distortion of each feature in $\mathbf{x}$ so that new models can be accurately learned from existing stored data points.

Suppose a data point $\mathbf{x}$ is composed of K features $x_1, x_2, \ldots, x_K$, and the ith feature x_i is an n_i-bit unsigned integer represented in binary, or a signed integer represented in two's complement with n_i magnitude bits and one sign bit. We let W be the total number of bits among all the uncoded feature vectors. For example, if all the features are unsigned, $W = \sum_{i=1}^{K} n_i$.

We wish to protect data points of this type against substitution errors produced by a binary symmetric channel (BSC) with crossover probability ϵ_{BSC}. Suppose we are allowed a budget of N bits per data point. In order to best maintain the predictive power of the algorithm, we will likely desire to protect features of higher importance more than features of lower importance. At a finer level of granularity, we may also wish to protect more important bits within a feature's binary representation more than less important bits for the same feature. Finally, since each feature is useful independently of the other features, the code should have some chance of decoding each feature, regardless of whether the entire feature vector is decodable.

Many popular error-correcting codes are not explicitly designed to address any of these desired properties. However, the simple repetition code presented in Section II has all of these properties. We can protect a bit b_1 of higher importance more than a bit b_2 of lower importance by simply repeating the b_1 more than b_2. Each feature bit is decoded independently of the other bits using maximum-likelihood decoding. More sophisticated codes that have a subset of the desired properties (such as variable strength codes) exist; however, in this work, we focus on non-uniform repetition coding as a first step.

Since we are employing repetition coding, we will optimize an objective function of our choice over all possible redundancy assignments subject to a budget N on the codeword length. We next discuss the choice of objective function.

Let $\mathbf{x}$ be a feature vector and x_i be the ith feature in $\mathbf{x}$. We can view x_i as an integer; however, x_i is stored in bits as $x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{in_i}$, where x_{ij} is the jth highest order magnitude bit in the binary representation. If the ith feature is a signed integer, let x_{i0} be the sign bit. We write $\mathbf{x}'$ for the noisy feature vector after it has been encoded using repetition coding, passed through a BSC, and decoded. Let $\mathbf{r}$ and $\boldsymbol{\epsilon}$ be two vectors of variables where r_{ij} is the number of times the bit x_{ij} is repeated in the codeword and ϵ_{ij} is the probability that a decoding error occurs for the bit x_{ij}. The elements r_{ij} appear in r in order of ij where $i_1 j_1 < i_2 j_2$ if $i_1 < i_2$, or if $i_1 = i_2$ and $j_1 < j_2$. The elements of $\boldsymbol{\epsilon}$ are ordered in the same way. Here, $j \in P_i = \{0, 1, ..., n_i\}$ if the ith feature is signed, and $j \in P_i = \{1, ..., n_i\}$ if the ith feature is unsigned. Clearly, ϵ_{ij} is a function of the underlying noise ϵ_{BSC} and the amount of redundancy used r_{ij}. Using the formula in (1), we have

$$\epsilon_{ij} = \sum_{k=\lceil r_{ij}/2 \rceil}^{r_{ij}} \binom{r_{ij}}{k} \epsilon_{\text{BSC}}^{k} (1 - \epsilon_{\text{BSC}})^{r_{ij}-k} \qquad (3)$$

for r_{ij} odd. The case of even r_{ij} uses equation (2)

A. Expected loss on predictions

One statistical measure of the predictive power maintained by redundancy assignment $\mathbf{r}$ is the expected loss on the predicted target value:

$$\mathbb{E}[\mathbb{E}[|\mathbf{x}^{\mathsf{T}}\mathbf{a} - \mathbf{x}'^{\mathsf{T}}\mathbf{a}| \mid \mathbf{x}]] = \sum_{\hat{\mathbf{x}}} \Pr(\hat{\mathbf{x}}) \mathbb{E}[|\hat{\mathbf{x}}^{\mathsf{T}}\mathbf{a} - \hat{\mathbf{x}}'^{\mathsf{T}}\mathbf{a}|].$$

For arbitrary $\mathbf{x}$, such an objective function requires us to know a prior distribution on $\mathbf{x}$.

One approach to redundancy assignment is to minimize this function directly. We refer to this model as M_1, and we can express it as:

$$\min_{\mathbf{r}} \mathbb{E}[\mathbb{E}[|\mathbf{x}^{\mathsf{T}}\mathbf{a} - \mathbf{x}'^{\mathsf{T}}\mathbf{a}| \mid \mathbf{x}]]$$
$$\text{such that:} \quad \sum_{i=1}^{K} \sum_{j \in P_i} r_{ij} = N \quad r_{ij} \geq 1.$$

Note that $\mathbf{r}$ is implicit in the objective function: each r_{ij} term determines ϵ_{ij} while each ϵ_{ij} determines a channel. The collective impact of these channels on $\mathbf{x}$ produces $\mathbf{x}'$.

A natural starting point for this problem setting is a uniform redundancy assignment. To get a sense of the improvement in $\mathbb{E}[\mathbb{E}[|\mathbf{x}^{\mathsf{T}}\mathbf{a} - \mathbf{x}'^{\mathsf{T}}\mathbf{a}| \mid \mathbf{x}]]$ compared to uniform redundancy assignment, we present the following example.

Example 1 *Consider a case where* $\mathbf{x}$ *is composed of two two-bit integers:* $\mathbf{x} = \begin{bmatrix}(x_{11} & x_{12}) & (x_{21} & x_{22})\end{bmatrix}$*, so that* $W = 2 \times 2 = 4$. *Let* $\epsilon_{BSC} = .2$*, and let the prior on* $\mathbf{x}$ *be a uniform distribution. The parameter* $\mathbf{a}$ *in the linear model is learned to be* $\mathbf{a} = \begin{bmatrix}100 & 50\end{bmatrix}^{\mathsf{T}}$*, and we select our total budget on codeword bits to be* $N = 12$. *Then, the optimal redundancy assignment found by* M_1 *is* $\begin{bmatrix}5 & 3 & 3 & 1\end{bmatrix}$*, and the expected loss on the prediction is* 37.78. *The uniform redundancy assignment is of course* $\begin{bmatrix}3 & 3 & 3 & 3\end{bmatrix}$*, and the expected loss on the prediction is* 42.23.

Clearly, the linear model's predictive power is better maintained using M_1's redundancy assignment. More pronounced improvements would occur with larger W and N values

While M_1 is a natural and promising model, it does not appear tractable, and there does not seem to be a convenient way to approximate it efficiently (by using, for example, a continuous relaxation). Also, we may not have access to an accurate prior distribution on $\mathbf{x}$. In addition to these issues, we observed that the model exhibits behavior that may not be desirable in some applications.

Example 2 *Say the features in* $\mathbf{x}$ *are two three-bit unsigned integers* ($W = 2 \times 3 = 6$). *Let* $\mathbf{x}$ *be defined by as*

$$\mathbf{x} = \begin{bmatrix}(x_{11} & x_{12} & x_{13}) & (x_{21} & x_{22} & x_{23})\end{bmatrix}^{\mathsf{T}}.$$

Suppose the parameter $\mathbf{a}$ *in the linear model is* $\mathbf{a} = \begin{bmatrix}100 & 80\end{bmatrix}^\mathsf{T}$. *Suppose this channel is also relatively noisy, with* $\epsilon_{BSC} = 0.3$. *Assume the prior on* $\mathbf{x}$ *is such that* $\mathbf{x} = \begin{bmatrix}(0 \ 1 \ 1) & (1 \ 1 \ 1)\end{bmatrix}^\mathsf{T}$ *with probability one. Let our total budget on codeword bits be* $N = 20$. *Then, the optimal redundancy allocation according to model* M_1 *can be computed to be* $\mathbf{r} = \begin{bmatrix}3 & 5 & 1 & 7 & 3 & 1\end{bmatrix}^\mathsf{T}$.

Observe that redundancy is assigned so that the most significant bit (the first bit) is less protected than the middle (second) bit of the first feature. The model intentionally makes the first feature more noisy than it needs to be! Intuitively, this happens because an error is likely to occur in one or more of the bits with value 1, $\{x_{12}, x_{13}, x_{21}, x_{22}, x_{23}\}$. To compensate for this error, the model makes the only 0-value bit x_{11} more noisy, increasing its expected value. This compensation helps the resulting prediction for $\mathbf{x}'$ match the noiseless prediction of $\mathbf{x}$. This behavior can also occur for a non-deterministic $\mathbf{x}$.

Intentionally making a feature more noisy may be useful for minimizing $\mathbb{E}[\mathbb{E}[|\mathbf{x}^\mathsf{T}\mathbf{a} - \mathbf{x}'^\mathsf{T}\mathbf{a}| \mid \mathbf{x}]]$, but it is counter-productive to preserving the integrity of the feature vector $\mathbf{x}$ itself, and could thus have negative effects on any future models learned using a stored value of $\mathbf{x}$. We could avoid such results by adding constraints such as $r_{ij_1} \geq r_{ij_2}$ for $j_1 < j_2$, to form model M_1', and while these constraints cut down the search space, the optimization remains intractable through exhaustive search.

Now consider a modified version of M_1 which we call $\hat{M}_1$:

$$\min_{\mathbf{r}} \mathbb{E}[\mathbb{E}[|\mathbf{x}^\mathsf{T}\mathbf{a} - \mathbf{x}'^\mathsf{T}\mathbf{a}| \mid \mathbf{x}]]$$

$$\text{s.t.} \quad \sum_{i=1}^{K} \sum_{j \in P_i} r_{ij} = N, \quad r \geq 0, \quad x'_{ij} = \alpha \text{ if } r_{ij} \text{ equals } 0,$$

where α if an arbitrary constant in $\mathbb{R}$ other than 1. This is essentially the same as M_1, except compression is allowed. Interestingly, this variation of the model is NP-hard:

Theorem 1 $\hat{M}_1$ *is NP-hard.*

Proof: The partition problem asks whether a set B of positive integers can be partitioned into two subsets whose elements sum to the same number. If the total sum of all numbers in B is S, then the partition problem is equivalent to asking whether B has a subset summing to $S/2$. The partition problem is known to be NP-complete.

We can show a many-to-one reduction from this problem to a variation of the subset sum problem, where the goal is to find a strict subset of a set of integers F such that the elements sum up to 0. In this particular variation we are guaranteed that all integers in F sum to 0. The partition problem reduces to this new problem by setting $F = B \cup \{-S/2, -S/2\}$. F has a strict subset summing to 0 if and only if B has a partition. This subset sum problem is obviously in NP and therefore NP-complete.

The reduction to $\hat{M}_1$ is as follows. Let $\epsilon_{\text{BSC}} = 0$, and the number of bits available be $|F| - 1$. Let $\mathbf{e}$ be a 0-1 vector denoting which features are encoded with one or more bits (some get 0 bits). Set the prior on $\mathbf{x}$'s to have probability 1 for the 1-vector. Let the linear regression's weight vector $\mathbf{a}$ consist of the numbers in F.

Now we have that the optimal solution has an expected error of 0 if and only if F has a subset sum zero if and only if B has a partition. The true output of the regression model on the 1-vector $\mathbf{x}$ is $\mathbf{a}^\mathsf{T}\mathbf{x} = \mathbf{0}$ (because the elements in F sum to 0). The expected output of the noisy regression model is $\mathbb{E}[\mathbf{a}^\mathsf{T}\mathbf{x}'] = (1-\alpha)\mathbf{a}^\mathsf{T}\mathbf{e}$. The error is 0 when $(1-\alpha)\mathbf{a}^\mathsf{T}\mathbf{e} = \mathbf{0}$ and thus $\mathbf{a}^\mathsf{T}\mathbf{e} = \mathbf{0}$. The vector $\mathbf{e}$ selects a subset of numbers in $\mathbf{a}$ and therefore in F that sum to 0. Thus, $\hat{M}_1$ is NP-hard. ■

Due to M_1's intractability, the potential unavailability of a prior, and unwanted behavior, we proceed to explore other strategies for assigning redundancy.

B. Individual distortion

One idea is to assign redundancy to a bit according to how much an error in that bit alone distorts the prediction, independently of other bits. The intuition behind this approach is that if a decoding error occurs in a small number of bits, it is desirable that each such bit's contribution to the error is minimized. The contribution to the prediction for bit x_{ij} is $|a_i|2^{(n_i - j)}$. The idea here is to limit the worst case magnitude of the distortion. In this sense, such a model provides an upper bound on the optimal value found by M_1.

One model with this property performs the following: minimize a weighted sum of ϵ_{ij}'s, where each ϵ_{ij} is weighted by the distortion of $\mathbf{a}^\mathsf{T}\mathbf{x}$ caused by an error in bit x_{ij} alone. The distortion is given by $|a_i|2^{(n_i - j)}$, and we refer to this value as the *influence* of bit x_{ij}. Observe that the influence of the sign bit for the ith feature is $|a_i|2^{n_i}$ since flipping the sign bit in two's complement representation changes the ith feature value by 2^{n_i}, regardless of the magnitude bit values. We refer to this model as M_2:

$$\min_{\boldsymbol{\epsilon}} \sum_{i=1}^{K} \sum_{j \in P_i} \epsilon_{ij} |a_i| 2^{(n_i - j)}$$

$$\text{such that:} \quad \sum_{i=1}^{K} \sum_{j \in P_i} r_{ij} = N, \quad r_{ij} \geq 1, \quad f(r_{ij}) = \epsilon_{ij},$$

where $f(r_{ij})$ is the probability of decoding error in (3).

To help justify this particular choice of objective function, observe that if all the errors are in the same direction with large probability, the upper bound on the objective value found by M_1 becomes tight.

Additionally this model minimizes $\mathbb{E}[\mathbb{E}[|\mathbf{x}^\mathsf{T}\mathbf{a} - \mathbf{x}'^\mathsf{T}\mathbf{a}| \mid \mathbf{x}]]$ for the following modification of the original BSC. Consider a modified channel C' where only a single bit is affected by noise. Suppose that each time $\mathbf{x}$ is passed through C', a random variable z takes on the value of a feature bit index ij. All feature bits except the zth bit are passed through the channel noiselessly. The encoding of the zth bit however, is passed through the original binary symmetric channel C. M_2 then

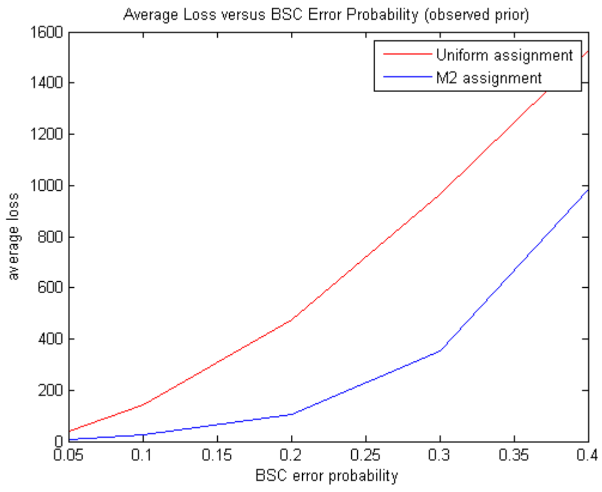

Fig. 3. Average loss for BSC crossover probabilities between 0.05 and 0.4 using the empirical prior on data points $\mathbf{x}$ for naive uniform redudancy assignment (red curve) versus M_2 redundancy assignment (blue curve).

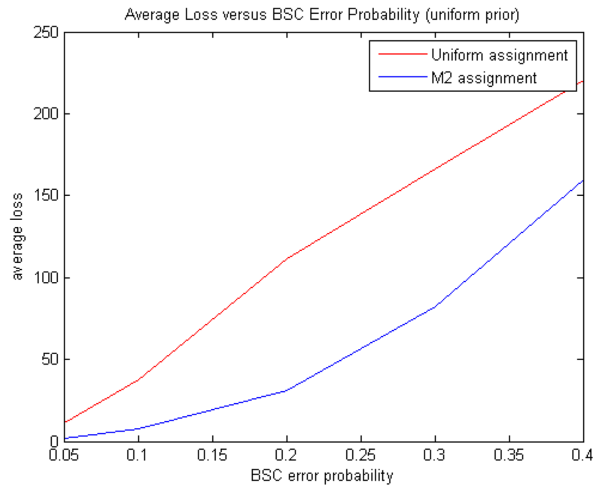

Fig. 4. Average loss for BSC crossover probabilities between 0.05 and 0.4 using a uniform prior on data points $\mathbf{x}$ for naive uniform redudancy assignment (red curve) versus M_2 redundancy assignment (blue curve).

gives the solution to the minimization of $W \times \mathbb{E}[\mathbb{E}[|\mathbf{x}^{\intercal}\mathbf{a} - \mathbf{x}'^{\intercal}\mathbf{a}| \mid \mathbf{x}]]$ under C' when z is uniformly distributed over the feature bits.

This model is convenient because a solution can be approximated using a convex relaxation as we will later show. Also, M_2 has the nice property that knowledge of the prior on $\mathbf{x}$ is not required, since the distortion incurred by only one bit error is the same, regardless of any original bit values. Finally, this model does not assign larger noise to more significant feature bits as M_1 did.

For Example 1, where $\mathbf{x}$ is composed of two two-bit features and there is a uniform prior on $\mathbf{x}$, M_2 produces the same redundancy assignment as M_1. The same improvement over uniform assignment is thus observed for M_2 as well.

Example 3 *We also tested M_2 on a large regression dataset from [12] where the data points are computers, each feature is a hardware specification, and the target value is the relative performance. The dataset has 13 features and 65 feature bits. For this test, $N = 195$ and $\epsilon_{BSC} = 0.3$. We used a continuous relaxation of M_2 to find a redundancy assignment. Since it is computationally intractable to calculate $\mathbb{E}[\mathbb{E}[|\mathbf{x}^{\intercal}\mathbf{a} - \mathbf{x}'^{\intercal}\mathbf{a}| \mid \mathbf{x}]]$ for such a large W value, we estimated it by calculating the average loss over many simulated examples. Using the prior on $\mathbf{x}$ from the data-set, we found that the average loss using M_2 was 351.99, while the average loss using uniform assignment was 964.43. This is nearly a three-fold improvement! Using a uniform prior on $\mathbf{x}$ we found that the average loss using M_2 was 82.27 while the average loss using uniform assignment was 166.46, which is roughly a two-fold improvement. Plots of the average loss versus the BSC crossover probability for the two prior assumptions are shown in Figures 3 and 4, respectively.*

In general, we can prove that M_2 never exhibits the unwanted behavior observed by M_1. The model always assigns redundancy to the bits in order of influence:

Lemma 1 *If bit $x_{i_1 j_1}$ has a higher influence than bit $x_{i_2 j_2}$ then the solution $\boldsymbol{\epsilon}^*$ to M_2 will satisfy $\epsilon^*_{i_1 j_1} \leq \epsilon^*_{i_2 j_2}$.*

Proof: Suppose that $\epsilon^*_{i_1 j_1} > \epsilon^*_{i_2 j_2}$ for the optimal assignment $\mathbf{r}^*$ to model M_2. Then $r_{i_1 j_1} < r_{i_2 j_2}$. Create a new assignment $\mathbf{r}'$ that is identical to $\mathbf{r}^*$ other than swapping the values of $r_{i_1 j_1}$ and $r_{i_2 j_2}$. This swaps the values of $\epsilon_{i_1 j_1}$ and $\epsilon_{i_2 j_2}$, and does not violate any constraints in the model M_2. Thus, in the new solution, $\epsilon'_{i_1 j_1} < \epsilon'_{i_2 j_2}$. However, we have that $|a_{i_1}|2^{(n_{i_1} - j_1)} > |a_{i_2}|2^{(n_{i_2} - j_2)}$. The objective value of the new solution is less than the objective value for $\mathbf{r}^*$, a contradiction. ■

The following corollary immediately follows:

Corollary 1 *For feature i, model M_2 will never assign $\epsilon_{ij_1} > \epsilon_{ij_2}$ for $j_1 < j_2$.*

Another desirable property satisfied by M_2 is the following. Since incrementing an odd r_{ij} by 1 does not decrease the probability of decoding error, we would prefer to avoid allocating an even number of bits to any feature, so that no redundancy bits are wasted. Of course, this is only possible if N, the total budget of bits, and W, the number of uncoded feature bits, are either both even or both odd:

Lemma 2 *Consider a binary symmetric channel (BSC) with crossover probability ϵ. Consider the length $r \geq 1$ repetition code of a single bit x. Suppose we use Maximum-Likelihood decoding. If $\frac{r}{2}$ errors occur when r is even, we then make a guess g about x's value. We choose g as 1 with probability .5. Then if r is odd, the probability of decoding error is equal to the probability of decoding error for $r+1$.*

Proof: The probability of decoding error for r is $\sum_{k=\frac{r+1}{2}}^{r}\binom{r}{k}\epsilon^k(1-\epsilon)^{r-k}$. The probability of decoding error for $r+1$ is

$$\sum_{k=\frac{r+3}{2}}^{r+1}\binom{r+1}{k}\epsilon^k(1-\epsilon)^{r+1-k}$$

$$+\Pr(x\neq g)\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}$$

$$=\sum_{k=\frac{r+3}{2}}^{r+1}\binom{r+1}{k}\epsilon^k(1-\epsilon)^{r+1-k}+.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}.$$

Decomposing the terms in the summation, we get

$$.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}$$

$$+\sum_{k=\frac{r+3}{2}}^{r+1}\binom{r}{k-1}\epsilon^k(1-\epsilon)^{r+1-k}+\binom{r}{k}\epsilon^k(1-\epsilon)^{r+1-k}$$

$$=.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}+\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+3}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$+\sum_{k=\frac{r+3}{2}}^{r}\binom{r}{k}\epsilon^k(1-\epsilon)^{r-k+1}+\binom{r}{k}\epsilon^{k+1}(1-\epsilon)^{r-k}$$

$$=.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}+\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+3}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$+((1-\epsilon)+\epsilon)\sum_{k=\frac{r+3}{2}}^{r}\binom{r}{k}\epsilon^k(1-\epsilon)^{r-k}$$

$$=.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}+\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+3}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$+\sum_{k=\frac{r+3}{2}}^{r}\binom{r}{k}\epsilon^k(1-\epsilon)^{r-k}.$$

For the two terms outside of the summation, we have that

$$.5\binom{r+1}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}+\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+3}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$=.5\binom{r}{\frac{r-1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}+.5\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}$$

$$+\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+3}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$=.5\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r+1}{2}}$$

$$+(.5(1-\epsilon)+\epsilon)\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

$$=\binom{r}{\frac{r+1}{2}}\epsilon^{\frac{r+1}{2}}(1-\epsilon)^{\frac{r-1}{2}}$$

The error probability is then $\sum_{k=\frac{r+1}{2}}^{r}\binom{r}{k}\epsilon^k(1-\epsilon)^{r-k}$. ■

Lemma 3 *If N is even and W is even, the optimal assignment $\mathbf{r}^*$ for model M_2 has no even entries. Similarly, if N is odd and W is odd, the optimal $\mathbf{r}^*$ for model M_2 has no even entries.*

Proof: Suppose N is even and W is even. Then $\mathbf{r}^*$ cannot have an odd number of even entries, because $\mathbf{r}^*$ would then have an odd number of odd entries, and N would be odd, giving a contradiction. If $\mathbf{r}^*$ has an even number of even entries, then form a new solution $\mathbf{r}'$ by subtracting one from all even entries of $\mathbf{r}$, and adding those subtracted ones to any entry in $\mathbf{r}'$. The objective function of M_2 for assignment $\mathbf{r}'$ is then lower than the objective function for $\mathbf{r}$. This is a contradiction and thus $\mathbf{r}^*$ cannot have an even number of even entries.

Now suppose N is odd and W is odd. Then $\mathbf{r}^*$ cannot have an odd number of even entries, because $\mathbf{r}^*$ would then have an even number of odd entries, and N would be even, giving a contradiction. If $\mathbf{r}^*$ has an even number of even entries, then form a new solution $\mathbf{r}'$ by subtracting one from all even entries of $\mathbf{r}$, and adding those subtracted ones to any entry in $\mathbf{r}'$. The objective function of M_2 for assignment $\mathbf{r}'$ is then lower than the objective function for $\mathbf{r}$. This is a contradiction and thus $\mathbf{r}^*$ cannot have an even number of even entries. ■

Lemma 1 proves that M_2 assigns redundancy according to the relative influence of each bit, avoiding the unwanted behavior observed in model M_1. Lemma 2 and lemma 3 prove that M_2 has an additional useful property.

Next, we explore the sensitivity of M_2's redundancy assignment to changes in the parameter $\mathbf{a}$ and the channel error ϵ_{BSC}. Table I shows the redundancy assignments for selected $\mathbf{a}$ and ϵ_{BSC}. Here, $\mathbf{x}$ has two two-bit unsigned integer features, and has a codeword budget of $N=16$. We observe that for higher ϵ_{BSC}, the difference between the assignments within features becomes more dramatic. Similarly, as the magnitudes of the a_i become more different, the differences between the assignments among the features become larger.

C. *M_2 approximation*

One of the main advantages of model M_2 is the existence of tractable approximations. This section is dedicated to such approximations. Since $\mathbf{r}$ is a vector of integers, M_2 is a discrete optimization problem. The number of possible redundancy assignments is lower bounded by $\binom{N+W-1}{W}$, which is prohibitively large if we wish to use an exhaustive search for non-trivial values of N and W. We do not know of any efficient algorithm to find the exact solution.

To approximate the optimal solution to M_2, we can relax r_{ij} to be continuous, and approximate the formula for ϵ_{ij} with a continuous function. We then put the problem into a convex form, obtain a solution using a convex solver, and perform rounding to obtain an integer assignment $\mathbf{r}$.

Our first task is to find a continuous approximation for $f(r_{ij})$. Observe that for even r_{ij}, $f(r_{ij})$ is equal to $f(r_{ij}-1)$ as seen in figure 5. We cannot hope to form a convex relaxation if this behavior is preserved in the approximation. We therefore choose to approximate ϵ_{ij} at odd values of r_{ij}, because M_2 assigns odd integers to r_{ij} in general. One

TABLE I
SENSITIVITY OF M_2 REDUNDANCY ASSIGNMENTS

a	[100 100]	[100 50]	[100 5]	[100 100]	[100 50]	[100 5]
ϵ_{BSC}	.3	.3	.3	.05	.05	.05
r	[5 3 5 3]	[9 3 3 1]	[9 5 1 1]	[5 3 5 3]	[5 3 5 3] or [5 5 3 3]	[5 5 3 3]

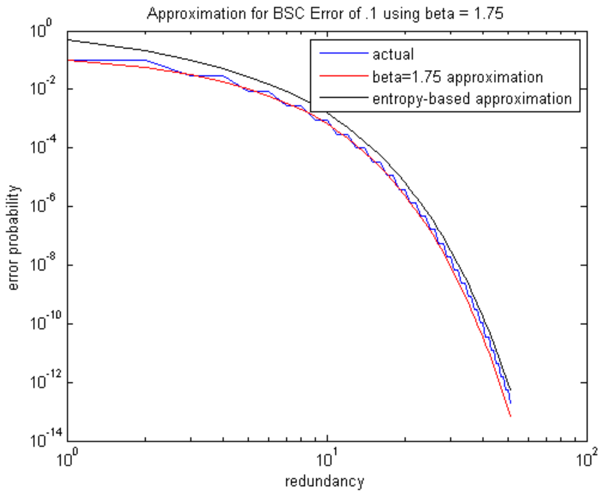

Fig. 5. Approximations to $f(r_{ij})$. The dark green curve is an entropy-based approximation, while the red curve is the approximation $\epsilon_{\text{BSC}}\beta^{(-r_{ij}+1)}$ for certain $\beta \in \mathbb{R}$.

conventional approximation of this function at odd values is $\binom{r_{ij}}{\lceil r_{ij}/2 \rceil}\epsilon_{\text{BSC}}^{\lceil r_{ij}/2 \rceil}(1-\epsilon_{\text{BSC}})^{\lfloor r_{ij}/2 \rfloor}$, the largest term in the summation. To approximate this as a continuous function, $\lceil r_{ij}/2 \rceil$ and $\lfloor r_{ij}/2 \rfloor$ can both be replaced with $r_{ij}/2$, and $\binom{r_{ij}}{r_{ij}/2}$ can be approximated as $\frac{2^{H(.5)r_{ij}}}{\sqrt{2\pi r_{ij}.5(1-.5)}}$ where $H(p)$ is the binary entropy function. The approximation of $f(r_{ij})$ is

$$\frac{2^{H(.5)r_{ij}}}{\sqrt{2\pi r_{ij}.5(1-.5)}}\epsilon_{\text{BSC}}^{.5r_{ij}}(1-\epsilon_{\text{BSC}})^{.5r_{ij}}.$$

Unfortunately, this approximation is very poor for small values of r_{ij}. This is unacceptable because it is these small values that we are most concerned with. We are interested in preserving $f(r_{ij})$ precisely for small values of r_{ij} because large changes occur in $f(r_{ij})$ when r_{ij} is small, and we often do not want to add a large amount of redundancy to the code, so that the rate loss is minimized. Furthermore, we cannot make the continuous problem convex if we use this approximation.

We found that $f(r_{ij})$ can be approximated very well with a function of the form $\epsilon_{\text{BSC}}\beta^{(-r_{ij}+1)}$ where $\beta \in \mathbb{R}$ depends on ϵ_{BSC}. The approximation can be made especially good for lower values of $\mathbf{r}$. Fortunately, this approximation allows us to put the relaxation of M_2 in convex form.

ϵ_{BSC}	β	ϵ_{BSC}	β	ϵ_{BSC}	β
.05	2.6	.1	1.75	.2	1.35
.3	1.14	.4	1.04		

TABLE II
β FOR VARIOUS ϵ_{ij}

An example of this approximation for $\epsilon_{\text{BSC}} = .1$ is shown in Figure 5. A table of β's for various ϵ_{ij} is given in Table II. Clearly, $\beta \geq 1$ for all probabilities ϵ_{BSC}.

Once we have found β to approximate $f(r_{ij})$ for ϵ_{BSC}, we have the relaxation of M_2 given by

$$\min_{\boldsymbol{\epsilon},\mathbf{r}} \sum_{i=1}^{K}\sum_{j\in P_i} \epsilon_{ij}|a_i|2^{(n_i-j)}$$

$$\text{such that:} \quad \sum_{i=1}^{K}\sum_{j\in P_i} r_{ij} = N, \quad \epsilon_{\text{BSC}}\beta^{(-r_{ij}+1)} = \epsilon_{ij}, \quad r_{ij} \geq 1.$$

We can express everything in terms of r, giving

$$\min_{\mathbf{r}} \sum_{i=1}^{K}\sum_{j\in P_i} \epsilon_{\text{BSC}}\beta^{(-r_{ij}+1)}|a_i|2^{(n_i-j)}$$

$$\text{such that:} \quad \sum_{i=1}^{K}\sum_{j\in P_i} r_{ij} = N, \quad r_{ij} \geq 1.$$

Taking the log of the objective function, we get an expression that resembles the log-sum-exp function which is known to be convex. Manipulating the objective function, we get

$$\min_{\mathbf{r}} \log_\beta\left(\sum_{i=1}^{K}\sum_{j\in P_i} \beta^{-r_{ij}+1+\log_\beta(\epsilon_{\text{BSC}}|a_i|2^{(n_i-j)})}\right).$$

Introducing a dummy variable $z_{ij} = -r_{ij} + 1 + \log_\beta(\epsilon_{\text{BSC}}|a_i|2^{(n_i-j)})$, we obtain the problem

$$\min_{\mathbf{r}} \log_\beta \sum_{i,j} \beta^{z_{ij}}$$

$$\text{such that:} \quad \sum_{i=1}^{K}\sum_{j=1}^{n_i} r_{ij} = N, \quad r_{ij} \geq 1,$$

$$z_{ij} = -r_{ij} + 1 + \log_\beta(\epsilon_{\text{BSC}}|a_i|2^{(n_i-j)}).$$

The objective function is now convex, and the constraints are linear. It is thus a convex optimization problem and can be solved efficiently with any convex optimization solver. Considering Example 2 where $\mathbf{x}$ is composed of two 3-bit features, and $\epsilon_{\text{BSC}} = .3$, the relaxation to model M_2 gives $\mathbf{r} =$

$\begin{bmatrix}1.0000 & 2.7064 & 7.9964 & 1.0000 & 1.0038 & 6.2934\end{bmatrix}^\intercal$. By rounding each entry to the nearest odd integer, we obtain our final approximation, $\hat{\mathbf{r}} = \begin{bmatrix}1 & 3 & 7 & 1 & 1 & 7\end{bmatrix}^\intercal$. We see that $\hat{\mathbf{r}}$ plugged into the objective function of M_2 gives 235.95, while the actual optimal solution $\begin{bmatrix}1 & 3 & 7 & 1 & 3 & 5\end{bmatrix}^\intercal$ to M_2 gives an objective value of 234.36. Clearly, the relaxation to M_2 was successful in finding an almost optimal solution to M_2 in this case. We also used it to find an assignment on the large dataset from [12] where we observed such dramatic performance over uniform redundancy assignment.

IV. Other applications and future work

We describe other applications. Our on-going work includes extensions to logistic regression, detection, and classification. We are also interested in other problem settings, where,

- The model parameters are also affected by noise and can be protected by redundancy bits from the coding budget.
- The training itself is performed in a noisy setting, so that the learned model is noisy.
- The code addresses more than one learning algorithm.

In each case, we must find an appropriate penalty function to optimize (as we did with models M_1 and M_2 for linear regression). As an example, for the naive Bayes classifier and the noiseless training/noiseless model setting we considered in this paper, the penalty function for classification must relate to the probability of the class being changed once noise is added. That is, we would like to maximize the probability that the classifier *preserves the class for noisy* $\mathbf{x}$' compared to $\mathbf{x}$:

$$\Pr\left\{\max_c \prod_i \Pr\{x_i|c\} = \max_c \prod_i \Pr\{x_i'|c\}\right\}. \quad (4)$$

Note that with a different penalty function and a different problem, the resulting coding scheme could be quite different from the one we derived for linear regression.

For binary classification, the probability (4) is equivalent to the *same decision probability* (SDP) [8], the probability of reaching the same decision $\Pr\{u \geq T|d\}$ if some hidden information $\mathbf{H}$ was used $\Pr\{u \geq T|d, \mathbf{H}\}$. Note that the probability in (4) can be viewed as the probability that the same decision (specifically the class variable) is reached when relying on either $\mathbf{x}$ or $\mathbf{x}'$. However, since x_i can be determined by x_i' and the indicator function ℓ_i for an error in the ith bit, we can write $\Pr\{x_i|c\}$ as $\Pr\{x_i'|c, \ell_i\}$. Then, we can view ℓ_i as the SDP hidden variable. We leave a computation (or approximation of) the SDP for future work; the following example shows that even a simple heuristic can improve the performance over a naive coding scheme.

Example 4 *We considered a naive Bayes classifier noiselessly trained on Fisher's Iris data set [11]. Here, the features are corrupted by additive white Gaussian noise (AWGN) with total variance between 0 and 2. We optimize the allocation of this noise into features. This task models coding without explicitly selecting a code (i.e., we view the encoder, channel, and decoder as a compound channel where the noise power is reduced.) The results were shown in the introduction in Fig.* 2.

The noise allocations were performed in two ways. The first (red curve) is equal strength among all four features. In the second allocation (blue curve), we computed a metric of information gain on the features in training. The information gain is defined by $H(c) - H(c|x_i)$, where H is the entropy function. The resulting information gain vector is $\begin{bmatrix}g_1 & g_2 & g_3 & g_4\end{bmatrix}^\intercal$. Here we allocated noise inversely proportional to this metric among all features, i.e., if the total noise power was V, the noise allocation for the ith feature $(1 \leq i \leq 4)$ had variance given by $\frac{1/g_i}{\sum_j 1/g_j} V$. In other words, noise is allocated inversely proportional to information gain among all four features.

V. Conclusion

In this paper we considered the problem of evaluating and optimizing noisy machine learning algorithms through channel coding. We provided an analysis of coding for the linear regression algorithm with two strategies. The first strategy directly optimizes our performance measure, but appears intractable, requires a prior distribution on the feature data, and exhibits some undesirable behavior. We introduced another principled strategy that we can approximate, does not require a prior distribution on the feature data, and does not exhibit unwanted behavior. We showed through simulation that our second strategy performs better than a naive approach on real data sets. We also introduced a number of problems for future study, including extending the current work for other applications and more challenging versions of the problem where the noise affects more than just the test data.

References

[1] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang, "Optimality guarantees for distributed statistical estimation," available: https://arxiv.org/pdf/1405.0782v2.pdf, 2014.

[2] T. S. Han and S. Amari, "Statistical inference under multiterminal data compression," *IEEE Trans. Info. Theory*, vol. 44, no. 6, pp. 2300-2324, Oct. 1998.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Machine Learning Research,* vol. 3, pp. 1157-1182, 2003.

[4] E. Kalapanidas, N. Avouris, M. Craciun, and D. Neagu, "Machine learning algorithms: a study on noise sensitivity," in *Proc. 1st Balcan Conference in Informatics*, Thessaloniki, 2003, pp. 356-365.

[5] O. Dekel and O. Shamir, "Learning to classify with missing and corrupted features, " in *Proc. 25th Int. Conf. on Machine Learning*, Helsinki, Finland, 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, 2006.

[7] A. Krause, H. B. McMahon, C. Guestrin and A. Gupta, "Robust submodular observation selection," *J. Machine Learning Research,* vol. 9, pp. 2761-2801.

[8] A. Choi, Y. Xue, and A. Darwiche, "Same-decision probability: A confidence measure for threshold-based decisions," *International Journal of Approximate Reasoning* vol. 53, pp. 1415-1428, Dec. 2012.

[9] 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.

[10] M. Ramoni and P. Sebastiani, "Robust Bayes classifiers," *Artificial Intelligence,* vol. 125, no. 1-2, pp. 209-226, Jan. 2001.

[11] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics,* vol. 7, pp. 179-188, 1936.

[12] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.