Abstract—In this article, we introduce new techniques to solve the nonlinear regression problem and the nonlinear classification problem. Our benchmarks suggest that our method for regression is significantly more effective when compared to classical methods and our method for classification is competitive. Our list of classical methods includes least squares, random forests, decision trees, boosted trees, nearest neighbors, logistic regression, SVMs and neural networks. These new techniques rely on convex variational formulations of the nonlinear regression and nonlinear classification problems. In the case of regression, we chose a function minimizing an energy functional plus the squared error of the predictions. In the case of classification, we chose a function minimizing an energy functional plus costs of misclassification. These convex variational formulations also provide information to perform dimensionality reduction and to study the dependencies between the variables of the problems. We also derive a notion of complexity for regression and classification problems. The method to find such minimizing functions turns out to be a simple quadratic optimization problem that can be solved efficiently. Here we present the methods in a way they can be easily understood by all practitioners without going into mathematical details.

Keywords—Regression, Nonlinear Regression, Classification, Nonlinear Classification, Variational Formulations, Optimization, Quadratic Programming, Machine Learning

I. INTRODUCTION

The regression problem has been studied for centuries and recently gained renewed attention due to the huge variety of applications it has in machine learning problems. When the function generating the data is linear, least squares is the most effective technique. On the other hand, when the function generating the data is nonlinear there are several computational techniques available to solve the problem. The most important ones are random forests, decision trees, boosted trees and neural networks. When the specific form of the function that generates the data is known, one may also try to use nonlinear optimization techniques.

The classification problem is more recent than the regression problem and also has several applications in machine learning problems. Support vector machines can be applied successfully when there is a plane that separates the groups in the training data. For the cases where such plane do not exist we can use the kernel trick or techniques such as decision trees, neural networks, logistic regression, random forests and boosted trees. In what follows, we try to give an overview of the principles of each of the classical techniques mentioned above independently of whether they are used for regression or classification.

The universal approximation theorem basically states that every continuous function on a compact set of $\mathbb{R}^n$ can be represented by a neural network. Such theorem indicates that neural networks can be powerful. However, the existing optimization methods to find the parameters of the neural networks are not convex. In addition, the architecture of the neural network is not easy to determine and requires human interference. Besides, neural networks are difficult and slow to train. They also need to be completely retrained as soon as the training set changes.

Decision trees are simple to understand and interpret, requires little data preparation and are considered to perform well. However, the problem of learning an optimal decision tree is known to be NP-complete. As a consequence, practical implementations of decision trees use greedy algorithms. Besides, the ability of decision trees to represent nonlinear functions is known to be limited. For instance, they cannot represent concepts such as parity, multiplexation and the logical operation exclusive or.

Random forests and boosted trees are ensemble learning techniques. They combine different decisions trees in an attempt to learn more complex nonlinear functions. Both algorithms are greedy. The training algorithm for random forests applies the general technique of bootstrap aggregating. The last technique consists in training different decision trees with random subsets of the training data. Boosted trees consist in learning consecutive decision trees that try to correct the mistakes of the previous ones.

Nearest neighbors is a technique that tries to give the answer based on the behavior of the training data near the point for which we need to give a prediction. Logistic regression and SVMs are similar in the sense that they try to find the parameters of a plane that is used to determine the regions in which the groups of the classification problems would be.

In the case that the form of the nonlinear function is known, one can try to adjust the coefficients of the nonlinear function to fit the data. Such techniques often have to deal with the existence of many local optima. In general, to use nonlinear optimization techniques one has to rely on humans to recognize the form of the underlying nonlinear function.

In this article we introduce new techniques to solve the nonlinear regression problem and the nonlinear classification problem. Our benchmarks suggest that our methods are more effective when compared to classical methods such as least squares, random forests, decision trees, boosted trees, nearest neighbors, logistic regression, SVMs and neural networks.
This new techniques rely on convex variational formulations of the nonlinear regression and classification problems. In the case of regression, we chose a function minimizing an energy functional plus the squared error of the predictions. In the case of classification, we chose a function minimizing an energy functional plus costs of misclassification. These convex variational formulations also provide information to perform dimensionality reduction and to study the dependencies between the variables of the problems.

A strong property of our methods is that the estimators (or classifiers) calculated with a training set $T_1$ are suboptimal estimators (or classifiers) for a training set $T_2$ provided that $T_1 \subset T_2$. This means that, with our methods, no computational effort is lost. We can keep updating the estimators and classifiers when new training data come.

In what follows, we call spectral regression the method introduced here for regression and spectral classification the method introduced here for classification. Each one of our methods have only one nonnegative parameter, which is $\lambda$. It reflects concepts such as bias and variance.

Roughly speaking, the closer $\lambda$ is to zero, the simpler the estimator (resp. classifier) generated by the spectral regression (resp. spectral classification). If $\lambda = 0$, the estimator (resp. the classifier) generated is zero. The larger lambda, the more likely the methods are to overfit. Here we do not provide a method to find the optimum values of $\lambda$.

We present the methods in a way they can be easily understood by all practitioners without going into mathematical details. As will be clear later, the methods need a procedure to generate columns for the underlying quadratic optimization problems used to solve the variational formulations. We provide a highly parallelizable brute force answer to the general column generation question. We also give a simplified strategy that is sufficient in many real situations.

The article is organized as follows. In Section II we give some preliminaries that are needed to understand the formulations given here. In Section III we introduce the variational formulation of the spectral regression and perform a series of transformations so that it can be solved by a quadratic optimization problem. In Section IV we give the formulation of the spectral classification and explain how it can be solved computationally.

In Section V we give a simplified strategy to generate columns for the quadratic optimization problems that appear. In Section VII we show how to study the dependencies of the variables of the problem in terms of the intensities of Fourier coefficients of the estimators and classifiers. In sections VIII and IX we show the benchmarks of our methods against the implementations of the classical methods for regression and classification provided by the Turi API.

II. Preliminaries

So that one can understand the convex variational formulations of the methods presented here, it is necessary to learn the multindex notation for derivatives and for exponentiation. It is also necessary to understand how the norm of a function and the norms of its derivatives measure its complexity. These are the topics of this section.

Take $f : (-0.5, 0.5)^d \rightarrow \mathbb{C}, f \in C^\infty$ and $\alpha \in \mathbb{N}^d$. Here we consider that $0 \in \mathbb{N}^d$. Denote by $\partial^\alpha f$ the partial derivative of $f$ with respect to $x_1$ taken $\alpha_1$ times. We denote by $\partial^\alpha f$ the sequence of derivatives $\partial^1 f, \partial^2 f, \ldots, \partial^d f$. As $f \in C^\infty$, $\partial^\alpha f$ does not depend on the order that the partial derivatives are taken. Let $v \in C^d$ and $\alpha \in \mathbb{N}^d$. We denote by $v^\alpha$ the product of the complex numbers $v_1, \ldots, v_d$. If $\alpha = 0$, then $\partial^\alpha f = f$.

We now give a brief explanation on energies of functions and how they can be used to implement the Occam’s razor in the context of nonlinear regression and nonlinear classification. First of all, the Occam’s razor is the principle that the simplest models that produce the right results are the ones most likely to generalize well to new situations.

Take an infinitely differentiable function $f : (-0.5, 0.5)^d \rightarrow \mathbb{C}$. Assume that the integral of $|f|^2$ over the entire open hypercube $(-0.5, 0.5)^d$ exists. By definition, the last integral is called the energy of $f$. The simplest function is the one equal to zero. If $f$ is continuous, its energy is zero if and only if $f$ is zero. If a function has values close to zero, its energy is also close to zero.

The concept of energy also can be explored for the partial derivatives of a function $f : (-0.5, 0.5)^d \rightarrow \mathbb{C}$ if $f$ has square integrable partial derivatives. Assume such property holds. If the energy of all partial derivatives of $f$ are small, then $f$ tends to be constant. On the other hand, if the energy of the derivatives is high, then the function tends to have more variations.

With such concepts in mind, it is simple to understand the relation between the spectral regression and the Occam’s razor. Using energy information about the derivatives of the functions in a suitable space, the spectral regression method chooses the simplest function that passes near the points in the training set with a cost of passing far away from the training points. Analogously, the spectral classification method chooses the simplest function that separates data of different groups.

We now turn to the crucial point that makes the ideas exposed above implementable. The space of square integrable functions is a Hilbert Space and has a geometrical structure very similar to $\mathbb{R}^n$. More precisely, we can measure the energies of general nonlinear functions in terms of their Fourier coefficients using the Parseval’s Identity. Moreover, the convergence of the Fourier series is exponentially fast under reasonable circumstances.

III. Spectral Regression

Assume we are given points $x_j \in \mathbb{R}^d, y_j \in \mathbb{R}$ for $j \in \{1, \ldots, J\}$. Without loss of generality, assume that $x_j \in [-0.25, 0.25]^d$. Consider given $\alpha_l \in \mathbb{N}^d$ for $l \in \{1, \ldots, L\}$. Define $H = \{ f : (-0.5, 0.5)^d \rightarrow \mathbb{C} : f \in C^{\infty} \cap L^2 \}$. Assume we are given $\lambda \geq 0$. The spectral regression technique consists in solving the following optimization problem:

\[
\begin{align*}
& \underset{s \in \mathbb{R}^J, f \in H}{\text{minimize}} & & \sum_{l=1}^L \| \partial^{\alpha_l} f \|^2 + \lambda \sum_{j=1}^J s_j^2 \\
& \text{subject to} & & \text{Re } f(x_j) = y_j + s_j
\end{align*}
\]  
(1)
There are some issues related with the optimization problem (1). Note that the vector space $H$ is not complete in the $L^2$ norm and if we change $H$ by the space of square integrable functions, the equalities in (1) do not make sense. However, for presentation purposes, we present (1) because we think it makes a good balance between the theoretically consistent formulation and the formulation that most people understand. We do not give the technical details associated with a consistent formulation of (1) in this article.

Assume that $f$ can be extended continuously to $[-0.5, 0.5]^d$ so that we can evaluate $f$ at the corners of the hypercube $[-0.5, 0.5]^d$. To write (1) in a tractable manner, we express the energy of the derivatives appearing in the objective function of (1) in terms of the associated Fourier coefficients. If $f(x) = 0$ for every $x \in \mathbb{R}^d$ that is a corner of the hypercube $[-0.5, 0.5]^d$, then we know that $\partial^{\alpha} f(k) = (2i\pi k)^{\alpha} \hat{f}(k)$.

Requiring that $f(x) = 0$ for every corner $x$ of the mentioned hypercube introduces an exponential number of constraints in (1). It can be seen experimentally that such constraints do not affect the solution of (1) in regions that are full of points. Motivated by the last observation, we use $\partial^{\alpha} f(k) = (2i\pi k)^{\alpha} \hat{f}(k)$ to compute the energies of the desired derivatives. Fix $f \in H$. Using Parseval’s Identity and the last formula we have:

$$\| \partial^{\alpha} f \|_2^2 = (2i\pi k)^{2\alpha} \sum_{k \in \mathbb{Z}^d} \| \hat{f}(k) \|^2.$$  

(2)

The advantage of writing the norms of the derivatives appearing in (1) in terms of the Fourier coefficients is that the space $L^2$ of square summable complex sequences is much more tractable than $H$. To introduce a more concrete formulation of (1) we define $\hat{H} = \{ f \in L^2 : f \in H \}$ and a polynomial $p(k) = \sum_{l=1}^{L} (2i\pi k)^{2\alpha_l}$ in $k \in \mathbb{Z}^d$ taking only positive values. Then, problem (1) becomes the following:

$$\begin{align*}
\text{minimize} & \quad \sum_{k \in \mathbb{Z}^d} \| \hat{f}(k) \|^2_2 p(k) + \lambda \sum_{j=1}^{J} s_j^2, \\
\text{subject to} & \quad \Re \sum_{k \in \mathbb{Z}^d} \hat{f}(k) e^{2\pi i k^T} = y_j + s_j.
\end{align*}$$  

(3)

As problem (3) has infinitely many variables, we are forced to choose some frequencies to solve (3) numerically. In suitable situations, the convergence of the Fourier series is exponential. This indicates that truncating problem (3) do not limit severely our ability to fit arbitrary nonlinear functions. To truncate problem (3), we consider given a finite number of frequencies $k_n \in \mathbb{Z}^d$ for $n \in \{1, ..., M\}$. The numerically tractable version of the spectral regression is the following:

$$\begin{align*}
\text{minimize} & \quad \sum_{n=1}^{M} \| \hat{f}(k_n) \|^2_2 p(k_n) + \lambda \sum_{j=1}^{J} s_j^2, \\
\text{subject to} & \quad \Re \sum_{n=1}^{M} \hat{f}(k_n) e^{2\pi i k_n^T} = y_j + s_j.
\end{align*}$$  

(4)

We now turn our attention to the polynomial $p(k)$. If we take $L = 1$ and $\alpha_1 = 0 \in \mathbb{N}^d$, then $p(k) = 1$. Such choice forces (4) to minimize the energy of the estimator of the data. In this case the estimator can have high derivatives. If we take $L = d + 1$ and $\alpha_1 = 0 \in \mathbb{N}^d$ and $\alpha_{l+1} = e_l \in \mathbb{N}^d$, then (4) tries to minimize the energy of the estimator and also the energy of its partial derivatives. In this case we have $p(k) = 1 + 4\pi^2 \| k \|_2^2$.

Other choices of sets $\{ \alpha_l : l = 1, ..., L \}$ are also possible. In this article we restrict ourselves to $p(k) = 1 + 4\pi^2 \| k \|_2^2$.

IV. SPECTRAL CLASSIFICATION

Assume again we are given points $x_j \in \mathbb{R}^d$ for $j \in \{1, ..., J\}$ and also disjoint sets $A, B$ whose union is $\{1, ..., J\}$. The sets $A, B$ tell which point belongs to which group. Without loss of generality, assume that $x_j \in [-0.25, 0.25]^d$. Consider given $\alpha_l \in \mathbb{N}^d$ for $l \in \{1, ..., L\}$. Define $H = \{ f : (-0.5, 0.5)^d \rightarrow \mathbb{C} : f \in C^\infty \cap L^2 \}$. Assume we are given $\lambda \geq 0$ and $\epsilon > 0$. The spectral classification technique consists in solving the following optimization problem:

$$\begin{align*}
\text{minimize} & \quad \sum_{l=1}^{L} \| \partial^{\alpha_l} f \|^2_2 + \lambda \sum_{j=1}^{J} s_j^2, \\
\text{subject to} & \quad \Re f(x_j) \leq -\epsilon + s_j, \quad j \in A, \\
& \quad \Re f(x_j) \geq \epsilon - s_j, \quad j \in B.
\end{align*}$$  

(5)

Applying the same transformations used in (1) we can also derive a computationally feasible form of (5). For this purpose, assume we are given frequencies $k_n \in \mathbb{Z}^d$ for $n \in \{1, ..., M\}$. Let $p(k) = \sum_{l=1}^{L} (2i\pi k)^{2\alpha_l}$ as in Section III. The numerical form of (5) is given by:

$$\begin{align*}
\text{minimize} & \quad \sum_{n=1}^{M} \| \hat{f}(k_n) \|^2_2 p(k_n) + \lambda \sum_{j=1}^{J} s_j^2, \\
\text{subject to} & \quad \Re \sum_{n=1}^{M} \hat{f}(k_n) e^{2\pi i k_n^T} \leq -\epsilon + s_j, \quad j \in A, \\
& \quad \Re \sum_{n=1}^{M} \hat{f}(k_n) e^{2\pi i k_n^T} \geq \epsilon - s_j, \quad j \in B.
\end{align*}$$  

(6)

The intuition over problem (6) is that the boundary that separates the two groups is the set $\{ x : f(x) = 0 \}$. This way, the spectral classification technique can be seen as a generalization of the support vector machine. In fact, as we show later, when the separation boundary is linear the spectral classification learns a linear boundary.

To deal with classification problems with $n > 2$ groups, the strategy is to learn $n - 1$ classifiers that are able to tell if a point $x$ belongs to a certain group $A$ or if $x$ does not belong to $A$. The readers may try other formulations for the nonlinear classification problem for more than two groups, for instance, splitting the image of the functions in more regions.

In this case the association between groups and regions becomes arbitrary. This arbitrary association may lead to separating functions that are not of minimum energy. This violates the Occam’s razor principle that says we should choose the simplest models.
V. GENERATING FREQUENCIES

Here we discuss the case of regression. The extension for classification follows the same direction. It is natural to ask if it is possible to automate the choice of the frequencies that should be considered in problem (4). To answer this question, we begin considering a simple case.

Suppose that the data we are trying to fit come from a separable function. It is trivial to see that the useful frequencies are the ones with at most one coordinate different of zero. Precisely, for a given constant $M \in \mathbb{N}$, the frequencies we need to consider are $\{\beta e_j\}$ with $l \in \{1, \ldots, d\}$ and $\beta \in \{-M, \ldots, M\}$. The amount of frequencies of the type $\{\beta e_j\}$ grows linearly with $d$.

If the function that generates the data we want to fit can be decomposed in functions depending only on combinations two by two of the variables of the problem, it is clear that the frequencies we need to consider are the ones with at most two coordinates different of zero. Depending on the value of $d$, this strategy can be used successfully because solving (4) is equivalent to solving a linear system and there are good large scale solvers for this purpose.

In real life large scale regression problems, the output of the function we want to fit is more likely to be the sum of other functions that depend only on subsets of variables of the original problem. In general, we can search for frequencies with at most $q$ entries different of zero that can contribute to the value of the objective function up to a certain tolerance. If we cannot find frequencies satisfying these conditions we increase $q$ until it reaches $d$.

The algorithm for frequency generation just described can be implemented using a highly parallel brute force algorithm as we explain now. Note that we can think of problem (4) as having all frequencies, but with additional constraints of the type $f(k) = 0$ for $k \neq k_n \ \forall n \in \{1, \ldots, M\}$. For simplicity, take $k_0 \neq k_n \ \forall n \in \{1, \ldots, M\}$.

Note that problem (4) is feasible and quadratic, then strong duality holds. Therefore, the dual variables of problem (4) provide useful information for sensitivity analysis with respect to the right-hand side of the constraints. Precisely, we want to know what is the effect of changing the right-hand side the constraint $f(k_0) = 0$ on the objective function of (4).

Such approach is justified by the fact that under suitable assumptions the derivative of the optimum value function with respect to the right-hand side of a constraint is equal to minus the associated dual variable.

We now give formulas for the dual variables $\nu_{\text{Re}}(k_0)$ and $\nu_{\text{Im}}(k_0)$ associated, respectively, with the real and imaginary parts of the constraint $f(k_0) = 0$. Let $\tilde{s}$ be the optimum primal value of $s$ of formulation (4). Using the optimality condition for convex problems, we can see that:

$$\nu_{\text{Re}}(k_0) = 2\lambda \sum_{j=1}^{J} \tilde{s}_j \cos(2\pi x_j^T k_0) \quad (7)$$

$$\nu_{\text{Im}}(k_0) = 2\lambda \sum_{j=1}^{J} \tilde{s}_j \sin(2\pi x_j^T k_0) \quad (8)$$

Assume we include a frequency $k_0 \in \mathbb{N}^d$ in (4). Note that we do not know a priori what are the signs of the values of the real and imaginary parts of $\hat{f}(k_0)$ at the optimum of the new problem (4). To find the frequencies that have the highest chances of improving our estimator, we need to maximize the absolute values of $\nu_{\text{Re}}(k_0)$ and $\nu_{\text{Im}}(k_0)$. The last task can be done in parallel.

We can identify a set of frequencies with high values of $|\nu_{\text{Re}}|$ and another set of frequencies with high values of $|\nu_{\text{Im}}|$. We can first solve (4) with frequencies with at most two coordinates different than zero. Then we begin adding new frequencies and resolving (4) using hot-starts.

The strategy described for frequency generation allows the possible users of the method to have computational gains by assuming that the outputs of the function generating the data are the sum of functions that only depend on subsets of variables of the original problem up to a certain size. Such assumptions may be realistic.

For very high dimensional nonlinear regression problems it may be worth it to project the data in lower dimensional subspaces using techniques such as singular value decomposition or principal component analysis. Such projections may kill some nonlinearities in the data. However, they create problems whose frequencies have lower dimensions, which may facilitate frequency generation strategies.

VI. DIMENSIONALITY REDUCTION

We can study dimensionality reduction by studying the intensities of the frequencies that we put in problem (4). Define $G_M = \{k \in \mathbb{Z}^d : \|k\|_{\infty} \leq M\}$ and suppose we take the lattice $G_M$ and put in (4). If the sum of norms of the Fourier coefficients associated with frequencies with the $i-th$ entry different of zero is small, we can conclude that the output of the training data does not depend on the $i-th$ variable. The generalization of this argument to the nonlinear classification problem is straightforward.

VII. THE DEPENDENCY HYPERGRAPHS

Besides providing tools to study dimensionality reduction, the spectral regression also provides tools to study the relationship between the variables of the problem. Let $X = \{1, \ldots, d\}$ be a set with indexes denoting the variables of the regression problem. Let $E^\beta$ be the set of all subsets of $X$ with $\beta > 0$ elements. We call the pair $(X, E^\beta)$ the dependency hypergraph of order $\beta$.

By definition, here we call by relation of order $\beta > 0$ a infinitely differentiable function with $\beta$ entries. For instance, $x_1^2 x_2 x_7$ is a relation of order 3 between the variables $x_1, x_2$ and $x_7$. Before continuing, assume we are given a square integrable function $f : (-0.5, 0.5)^d \rightarrow \mathbb{C}$ denoting the nonlinear function generating the output of the regression problem under consideration.

We now consider sets of frequencies induced by the hyper-edges belonging to $E^\beta$ as follows. Take $e^{\beta} \in E^\beta$. Note that $e^{\beta}$ is a set. Define $I_{e^{\beta}} = \{k \in \mathbb{Z}^d : k_i = 0 \ \forall i \in X - e^{\beta}, \ k_i \neq 0 \ \forall i \in e^{\beta}\}$. Note that if $e^{\beta}_1, e^{\beta}_2 \in E^\beta$ and $e^{\beta}_1 \neq e^{\beta}_2$, then $I_{e^{\beta}_1} \cap I_{e^{\beta}_2} = 0$. Note also that $0 \notin I_{e^{\beta}}$ $\forall e^{\beta} \in E^\beta$. 
For each $e_{\beta} \in E^{\beta}$, we can associate a number $c_{e_{\beta}}$ that is the sum of the square norms of the Fourier coefficients $\hat{f}(k)$ associated with frequencies $k$ that belong to $I_{e_{\beta}}$. By Parseval’s Identity, we can conclude that the sum of $c_{e_{\beta}}$ for all $\beta$ and $e_{\beta} \in E^{\beta}$ equals the energy of $f$ minus $\| \hat{f}(0) \|^2$. We cannot associate the intensity $\| \hat{f}(0) \|^2$ with a specific subset of variables.

Take $\beta$ satisfying $1 \leq \beta \leq d$. Let $c_{\beta}$ be the sum of $c_{e_{\beta}}$ for all $e_{\beta} \in E^{\beta}$. By Parseval’s Identity, the sum of all $c_{\beta}$ equals the energy of $f$ minus $\| \hat{f}(0) \|^2$. The numbers $c_{\beta}$ quantify how much of the output of the regression problem is given by relations of order $\beta$ between the variables of the problem.

The numbers $c_{\beta}$ give a measure of complexity for regression problems. This is intuitive because the higher the influence of relations of high order, the harder it is to generate frequencies that produce high quality estimators. The generalization of these ideas for nonlinear classification problems is straightforward.

VIII. Regression Benchmarks

We now present our benchmarks for the regression problem. The implementations of this project were done using the Julia programming language. All optimization problems were solved using mainly the packages JuMP and Convex.

The instances used in the experiments are shown in Table I. The error of the predictions shown in Table IV were measured using Monte Carlo simulations to calculate the expected value of the square error of the prediction.

In Figures 1 and 2 we see the results of the spectral regression method showing overfitting due to a high value of $\lambda$. In Table IV we see that the spectral regression presented inferior performance in linear instances when compared to the least squares method.

The benchmarks of Table V correspond to the instances of Table II. The corresponding images are shown in Figure 3. Note that the spectral regression works as an interpolator for instances with small number of points.

IX. Classification Benchmarks

In this section we present the benchmarks for the spectral classification method. Our implementation used the Julia programming language and the optimization packages JuMP and Convex. To compute the performance of the methods we used half of the points in the instances as training data and half of the points as test data.

The instances used are shown in Table III and the performances of the spectral classification and of the classical methods are shown in Table VI. We can observe that the spectral classification had a competitive performance. The Figures 4 and 5 show the boundaries learned by the spectral classification.

We made also some experiments with linear boundaries. In these experiments, the spectral classification performed slightly worse than the best classical methods. As we already mentioned, the spectral classification contains no bias in the form of the separation boundary besides the ones given by lambda and the energy minimization. This is not the case of the classical methods. We attribute this worse performance to the bias and to the small amount of data.

X. Conclusion

In this work we presented convex variational formulations for the nonlinear regression and nonlinear classification problems. Our benchmarks suggest that our methods are the most effective and general. More benchmarks and developments are clearly necessary to make the methods work in high dimensional problems.

XI. Future Works

We have at least two main branches to continue working in the ideas presented here. The first one is theoretical and the second one is computational. The theoretical work to be done consists in proving convergence theorems and studying convergence rates for the methods. It may also be worth connecting the formulations presented here with the theory of inverse problems.

The computational part consists in developing methods for frequency generation in high dimensions. It also may consist in developing special purpose methods to solve the optimization problems introduced here. We also may try to develop dimensionality reduction techniques based in Section VI or techniques to estimate the complexity of a given nonlinear regression problem based in the ideas presented in Section VII.

Acknowledgment

The author wants to thank his family for being patient with his absence and dedication with this project. Specially, he wants to thank his mother for all his life long support. The author of this article wants to thank his friend Pedro Ivo for reviewing this article and giving helpful comments. He also wants to thank professor Benar Svaiter for some theoretical considerations and Mario Veiga for suggesting the Turi API for benchmarks.

General Comments

The author of this article is not a specialist in machine learning and developed this project mostly for passion. This project was developed without financial support during the nights and weekends. It is also the first individual work of the author, who apologizes for all imperfections in the text. He takes all the responsibility for any serious mistakes that may exist in this text. Newer versions of this article will come. The reader with suggestions or doubts can send the author an email that will be answered as soon as possible.
REFERENCES


Pedro Borges received a master’s degree in Applied Mathematics from Brazil’s National Institute for Pure and Applied Mathematics (IMPA) with a specialization in fluid dynamics. At IMPA he advanced the numerical methods for dynamical systems with a thesis entitled Optimization Methods for Locating Heteroclinic Orbits. He has experience with both mathematics and computer science. Now he is working with energy planning at PSR.
Fig. 1: In these figures we see the instances inst1d01c, inst1d02c, inst1d03c, inst1d04c, inst1d05c and inst1d06c of table I. For these figures we used $\lambda = 3000$. This is the reason the results appear to be overfitting.
Fig. 2: In these figures we see the instances inst2d01c, inst2d02c, inst2d03c, inst2d04c, inst2d05c and inst2d06c of table I. For these figures we used $\lambda = 3000$. Note that the instance inst2d02c do not depend on one of the variables of the problem and that the method is able to perceive.
Fig. 3: In these figures we see that the spectral regression method works as an interpolator when the amount of data is small. Looking at the figures above one may imagine that the convergence properties of the spectral regression method are good. This is subject of future research. The instances above correspond to the instances inst1d01b, inst1d02d, inst1d03c, inst1d04a, inst1d05c and inst1d06a of Table II. For these figures $\lambda = 1000$. 
Fig. 4: In the left we see the instances inst2d01c, inst2d02c, inst2d03c and in the right the instances inst2d01cn, inst2d02cn and inst2d03cn of Table III. The points in black are an estimate for the separation boundary. The points in blue and red belong to different groups. The instances in the left do not have noise and the instances in the right have noise. In this figures we see that the spectral classification generalizes the support vector machine. We see that some boundaries are curved. This is a consequence of the noise and of the absence of bias in the method. Such curved boundaries are the ones with minimum energy. For these figures $\lambda = 2$. 
Fig. 5: In the left we see the instances inst2d04c, inst2d05c, inst2d06c and in the right the instances inst2d04cn, inst2d05cn and inst2d06cn of Table III. The points in black are an estimate for the separation boundary. The points in blue and red belong to different groups. The instances in the left do not have noise and the instances in the right have noise. We see that the spectral classification chooses to misclassify some points in order to get simpler boundaries. Note that the spectral classification method do not have any bias besides the ones given by $\lambda$ and by the Occam’s razor principle. This absence of external bias makes the boundaries be a product only of the data in the instances. For these figures $\lambda = 2$. 


TABLE I: In this table we can see the functions used in the experiments to evaluate the performance of the spectral regression method. We used frequencies with at most two coordinates different than zero. As the reader see, this is sufficient to represent the nonlinearity in all instances. Note that the instances can be related to the same nonlinear function, but with a different number of training points. The variance of the noise can change.
<table>
<thead>
<tr>
<th>Instance</th>
<th>Function</th>
<th>Number of Points</th>
<th>Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst01d01a</td>
<td>$x_1$</td>
<td>3</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d01b</td>
<td>$x_1$</td>
<td>5</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d01c</td>
<td>$x_1$</td>
<td>8</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d02a</td>
<td>$x_1^2$</td>
<td>4</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d02b</td>
<td>$x_1^2$</td>
<td>6</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d02c</td>
<td>$x_1^2 + 5x_1$</td>
<td>9</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d03a</td>
<td>$e^{-100x_1^2}$</td>
<td>4</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d03b</td>
<td>$e^{-100x_1^2}$</td>
<td>10</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d03c</td>
<td>$e^{-100x_1^2}$</td>
<td>15</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d04a</td>
<td>$\sin(-100x_1^2)$</td>
<td>15</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d04b</td>
<td>$\sin(-100x_1^2)$</td>
<td>30</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d04c</td>
<td>$\sin(-100x_1^2)$</td>
<td>50</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d05a</td>
<td>$\sin(-100x_1^2)$</td>
<td>15</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d05b</td>
<td>$\sin(-100x_1^2)$</td>
<td>25</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d05c</td>
<td>$\sin(-100x_1^2)$</td>
<td>55</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d06a</td>
<td>$\sin(-100x_1^2) + 3x_1$</td>
<td>20</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d06b</td>
<td>$\sin(-100x_1^2) + 3x_1$</td>
<td>50</td>
<td>$N(0,0)$</td>
</tr>
<tr>
<td>inst01d06c</td>
<td>$\sin(-100x_1^2) + 3x_1$</td>
<td>70</td>
<td>$N(0,0)$</td>
</tr>
</tbody>
</table>

TABLE II: In this table we show some instances used to evidence the fact that the spectral regression may work as an interpolator if the amount of data is small. These instances also evidentiate that the generalization ability of the spectral regression method are higher than the ones of the classical methods. The last affirmations are justified in Table V and in Figure 3.
TABLE III: In this table we can see the boundaries used to generate the nonlinear classification instances to produce the benchmarks for the spectral classification method. All instances are of two dimensions. For the same boundary we considered different numbers of points. We also considered instances with the same boundary but with and without the presence of noise.
TABLE IV: In this table we show the average value of the squared error of the predictions given by the spectral regression and by the classical methods for regression. In the first line of the table we used SE for spectral estimator, LS for least squares, BT for boosted trees, DT for decision trees, RF for random forest and NN for neural networks. The neural network predictions by the classical methods provide less stable results: they can give good results as well as they can give not so good results. In these experiments we used \( \lambda = 3000 \) for the spectral regression method. The expectral regression almost always ranks the best. The classical methods provide more stable results: they can give good results as well as they can give not so good results. In these experiments we used \( \lambda = 3000 \) for the spectral regression method.
### TABLE V: In this table we show the average value of the squared error of the predictions given by the spectral regression and by the classical methods for regression. In the first line of the table we used SE for spectral estimator, LS for least squares, BT for boosted trees, DT for decision trees, RF for random forest and NN for neural networks. The neural network predictions used one fully connected hidden layer. For these experiments $\lambda = 1000$. The numbers in bold indicates that the spectral regression had inferior performance comparatively to the least squares method.

<table>
<thead>
<tr>
<th>Instance</th>
<th>SE</th>
<th>LS</th>
<th>BT (d=1)</th>
<th>BT (d=2)</th>
<th>BT (d=3)</th>
<th>DT (d=3)</th>
<th>RF (d=3)</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst1d01a</td>
<td>0.0140472</td>
<td>2.52391e-06</td>
<td>0.0971879</td>
<td>0.0971879</td>
<td>0.0971879</td>
<td>0.160559</td>
<td>0.0307104</td>
<td>0.0214209</td>
</tr>
<tr>
<td>inst1d01b</td>
<td>0.000410417</td>
<td>1.33855e-07</td>
<td>0.0979259</td>
<td>0.0979259</td>
<td>0.0979259</td>
<td>0.157204</td>
<td>0.0254445</td>
<td>0.0195082</td>
</tr>
<tr>
<td>inst1d01c</td>
<td>0.0101425</td>
<td>2.09423e-07</td>
<td>0.0694154</td>
<td>0.0694154</td>
<td>0.0694154</td>
<td>0.132935</td>
<td>0.0215232</td>
<td>0.0252024</td>
</tr>
<tr>
<td>inst1d02a</td>
<td>8.6932e-05</td>
<td>0.000364702</td>
<td>0.0760952</td>
<td>0.0760952</td>
<td>0.0760952</td>
<td>0.132311</td>
<td>0.0124022</td>
<td>0.000496116</td>
</tr>
<tr>
<td>inst1d02b</td>
<td>0.000289642</td>
<td>3.804e-05</td>
<td>0.0658639</td>
<td>0.0658639</td>
<td>0.0658639</td>
<td>0.124665</td>
<td>0.0254445</td>
<td>0.000496116</td>
</tr>
<tr>
<td>inst1d03a</td>
<td>0.0101425</td>
<td>3.3804e-05</td>
<td>0.0658639</td>
<td>0.0658639</td>
<td>0.0658639</td>
<td>0.124665</td>
<td>0.0254445</td>
<td>0.000496116</td>
</tr>
<tr>
<td>inst1d03b</td>
<td>0.000146113</td>
<td>0.000620725</td>
<td>0.084082</td>
<td>0.084082</td>
<td>0.084082</td>
<td>0.138485</td>
<td>0.0222611</td>
<td>0.00129108</td>
</tr>
<tr>
<td>inst1d03c</td>
<td>0.00010912</td>
<td>0.00062162</td>
<td>0.060554</td>
<td>0.060554</td>
<td>0.060554</td>
<td>0.117214</td>
<td>0.0228218</td>
<td>0.00133923</td>
</tr>
<tr>
<td>inst1d04a</td>
<td>1.79198e-05</td>
<td>0.000538638</td>
<td>0.060029</td>
<td>0.060029</td>
<td>0.060029</td>
<td>0.117214</td>
<td>0.00186224</td>
<td>0.000108626</td>
</tr>
<tr>
<td>inst1d04b</td>
<td>3.4285e-05</td>
<td>0.129028</td>
<td>0.105667</td>
<td>0.105667</td>
<td>0.105667</td>
<td>0.136773</td>
<td>0.0257804</td>
<td>0.116373</td>
</tr>
<tr>
<td>inst1d04c</td>
<td>3.13563e-05</td>
<td>1.406699</td>
<td>0.102813</td>
<td>0.102813</td>
<td>0.102813</td>
<td>0.136773</td>
<td>0.0257804</td>
<td>0.116373</td>
</tr>
<tr>
<td>inst1d05a</td>
<td>0.296639</td>
<td>0.490375</td>
<td>0.535217</td>
<td>0.441235</td>
<td>0.485623</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
<tr>
<td>inst1d05b</td>
<td>0.053069</td>
<td>0.560190</td>
<td>0.573202</td>
<td>0.297858</td>
<td>0.297858</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
<tr>
<td>inst1d05c</td>
<td>0.060605</td>
<td>0.414362</td>
<td>0.446742</td>
<td>0.32088</td>
<td>0.32088</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
<tr>
<td>inst1d06a</td>
<td>0.0918627</td>
<td>0.415288</td>
<td>0.425638</td>
<td>0.382475</td>
<td>0.382475</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
<tr>
<td>inst1d06b</td>
<td>0.0388343</td>
<td>0.439354</td>
<td>0.388799</td>
<td>0.334247</td>
<td>0.334247</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
<tr>
<td>inst1d06c</td>
<td>0.0259211</td>
<td>0.405711</td>
<td>0.430138</td>
<td>0.382416</td>
<td>0.382416</td>
<td>0.600283</td>
<td>0.335334</td>
<td>0.461838</td>
</tr>
</tbody>
</table>
observe that the spectral classification has a superior performance when there is no noise in the instances. The numbers in bold stand for depth. The predictions using neural networks used an one layer perpeptron network. We can observe that the spectral classification has a superior performance when there is no noise in the instances. The numbers in bold indicate experiments for which the spectral classification had strictly inferior performance. In these experiments we used $\lambda = 2$ for the spectral classification method.

TABLE VI: In this table we can compare the performance of the spectral classification method against the classical methods.