WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Physics

Dissertation Examination Committee:
Zohar Nussinov, Chair
Anders Carlsson
Joseph O'Sullivan
Ralf Wessel
Li Yang

Physics-inspired Replica Approaches to Computer Science Problems

by

Bo Sun

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2017

Saint Louis, Missouri

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I am grateful to my advisor Professor Zohar Nussinov for his helpful guidance through the last five years. No single section of this dissertation would be possible without his guidance and supervision. I also appreciate the time and effort of my faculty mentoring committee members Professor Anders Carlsson and Li Yang. This thanks extend to my dissertation examination committee Professor Ralf Wessel and Joseph O'Sullivan. I wish to thank the group-mates Tahereh Mazaheri, Nicholas Weingartner, Seyyed Vaezi and Patrick Chao for many useful conversations and time spent together. I would like to take a moment a say thank you to my spouse Yadong Xu and my daughter Victoria Sun for their accompany and support. Finally, I would like to thank my parents who are always with me and supporting me through my life.

Bo Sun, Washington University, August 2017

ABSTRACT OF THE DISSERTATION

Physics-inspired Replica Approaches to Computer Science Problems

by

Bo Sun

Doctor of Philosophy in Physics

Washington University in St. Louis, 2017

Professor Zohar Nussinov, Chair

We study machine learning class classification problems and combinatorial optimization problems using physics inspired replica approaches. In the current work, we focus on the traveling salesman problem which is one of the most famous problems in the entire field of combinatorial optimization. Our approach is specifically motivated by the desire to avoid trapping in metastable local minima-a common occurrence in hard problems with multiple extrema. Our method involves (i) coupling otherwise independent simulations of a system (replicas) via geometrical distances as well as (ii) probabilistic inference applied to the solutions found by individual replicas. In particular, we apply our method to the well-known k-opt algorithm and examine two particular cases-k = 2 and k = 3. With the aid of geometrical coupling alone, we are able to determine for the optimum tour length on systems up to 280 cities (an order of magnitude larger than the largest systems typically solved by the bare k = 3 opt). The probabilistic replica-based inference approach improves k - opt even further and determines the optimal solution of a problem with 318 cities. In this work, we also formulate a supervised machine learning algorithm for classification problems which is called Stochastic Replica Voting Machine (SRVM). The method is based on the representations of known data via multiple linear expansions in terms of various stochastic functions. The algorithm is developed, implemented and applied to a binary and a 3-class classification problems in material science. Here, we employ

SRVM to predict candidate compounds capable of forming cubic Perovskite structure and further classify binary (AB) solids. We demonstrated that our SRVM method exceeds the well-known Support Vector Machine (SVM) in terms of accuracy when predicting the cubic Perovskite structure. The algorithm has also been tested on 8 diverse training data sets of various types and feature space dimensions from UCI machine learning repository. It has been shown to consistently match or exceed the accuracy of existing algorithms, while simultaneously avoiding many of their pitfalls.

Chapter 1

Introduction

1.1 The replica approaches

As will be made lucid in this work, in various problems, the term "replica" will allude to an individual solver/solution. We focused on two main problems where we applied our physics-inspired replica approaches. One is the famous traveling salesman problem, the other is supervised machine learning class classification problem. The motivation behind our replica approach was triggered by an anthropological principle known as "wisdom of the crowds". That is, a crowd or collection of solvers may solve a problem far more accurately than a single solver (or human). Fig. 1.1 illustrates how the replica approach works in general. In that figure, each sphere represents a single solver ("replica") for a specific problem which has its own configuration in the parameter space. Usually a single solver might get trapped in a local minimum or has some bias for the actual result. The collection of many solvers, however, might overcome the drawback described above and then help to find the global optimal solution and/or the most accurate result.

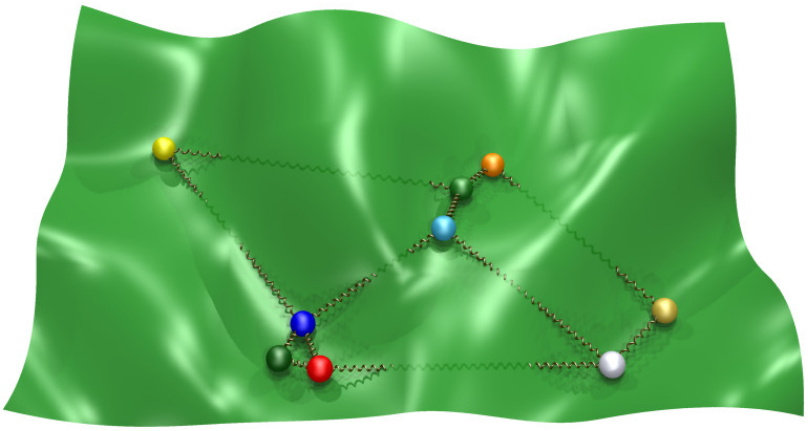


Figure 1.1: (Color online.) Coupled replicas in a high dimensional energy landscape. The springs schematically represent the tendency of replicas to collectively interact with one another when veering towards viable minima.

1.2 Main results of this dissertation

My research focused on three applications of applying Physics-inspired replica approach. Most of the new results are listed as follow: (a) An interacting replica approach to the traveling salesman problem. The traveling salesman problem is an NP-hard problem, and as such, no algorithm has been discovered which can solve it in polynomial time. We demonstrated that with the coupling of geometrical distance and probabilistic inference between different replicas we are able to solve optimally up to 318 cities. We further find that our algorithm can be capable of coupled with more sophisticated local minimization technique thus a better optimization results can be expected. (b) A stochastic Replica-based voting Machine Algorithm for supervised learning(SRVM). We formulated a supervised machine learning class classification algorithm in general. We developed, implemented and applied the algorithm on many benchmarks from UCI machine learning repository [1] and found that our algorithm match or exceed the accuracy of the existing state-of-art algorithm. (c) Stochastic Replica voting machine prediction of stable Perovskite and binary alloys. Here we employ SRVM to predict candidate compounds capable of forming cubic Perovskite structure and further classify binary(AB) solids. We determined the boundary in feature space delineating the requisite condition for material formation.

In below sections, I will briefly elaborate on these.

1.3 Traveling Salesman Problem

Combinatorial optimization problems [2] often possess a relatively large number of locally optimal pseudo-solutions, similar to the abundance of metastable energy states in complex physical systems. This can make determination of the global optimum difficult, especially for heuristic algorithms which attempt to optimize a cost function locally by iteratively perturbing the parameters, testing the resulting change in the cost function, and allowing the state change if the cost function is decreased or some other conditions are satisfied.

The Traveling Salesman Problem (TSP) is one of the most famous problems in the entire field of combinatorial optimization. It is often used as a benchmark for testing new optimization approaches. The TSP is an NP-hard problem, and as such, no algorithm has been discovered which can solve it in polynomial time. The problem is defined as follows:

Given a set of N cities, find the shortest tour which visits each city exactly once and returns to the starting city.

In the current work, we present new optimization methods based upon the concepts of (i) geometrical distance coupling (GDC) (see Fig. 1.2) and (ii) probabilistic inference amongst independent replicas (see Fig. 1.3). We apply these tools to the Traveling Salesman Problem. We demonstrate that while a single replica can do quite poorly in solving a challenging problem, via the use of (i) and (ii) above, the *ensemble* of replicas can address much harder problems.

1.4 Stochastic Replica-Based Voting Machine Methods

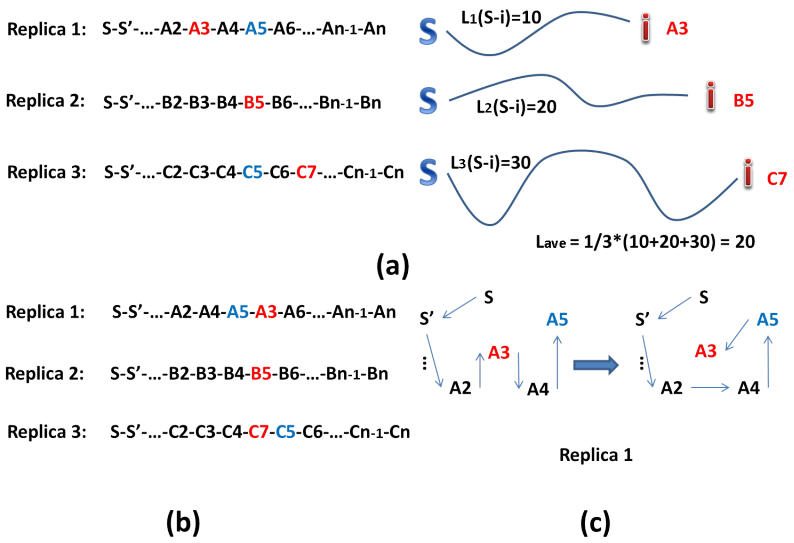


Figure 1.2: (Color online.) An illustration of the geometric coupling between replicas. The outcome of this basic coupling is that a given city (node) i is moved to a position averaged over all replicas. In this example, there are three replicas. The link SS' is common to all replicas. S is a "standard city" that is used to calibrate distances (see Appendix). This city is chosen at the beginning (by symmetry the choice of this city is immaterial). (a) Three relevant replicas sharing a common edge S-S'. i is the randomly chosen city. The designations A3, B5, and C7 represent the same city i as it appears in replica 1, replica 2 and replica 3 respectively. In these three individual replicas, the tour length between S and i are 10, 20, and 30 respectively. After averaging, the city i will be moved to cities with a distance of 20 away from S in all three replicas. A5, B5 and C5 are the target city where A3, B5 and C7 will be inserted (see Appendix). (b) Updated replica configurations after inserting city i into the target location. (c) A graphic depiction of the change between the initial (a) and final (b) replicas before and after the move of city i in replica 1.

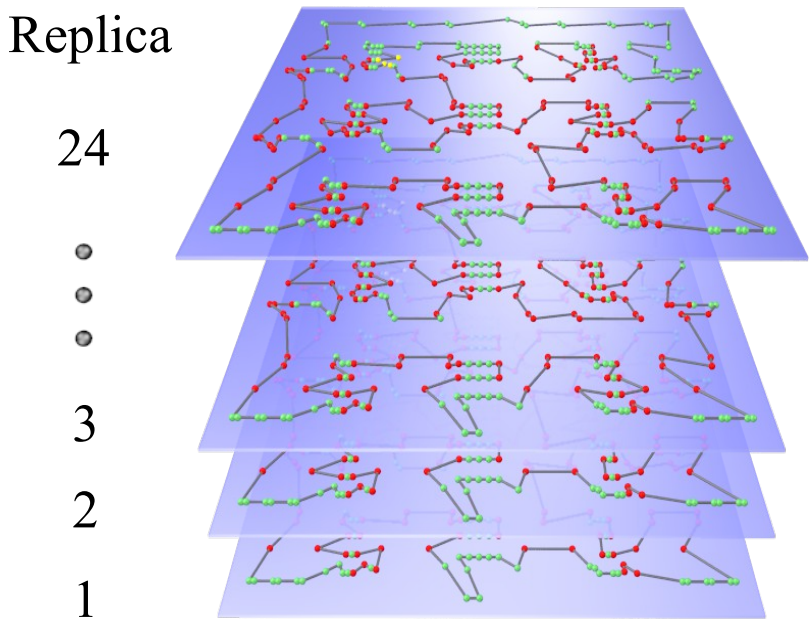


Figure 1.3: (Color online.) Inference between replicas for Problem lin318. We define a "bubble" as a set of nodes where the neighbor cities differ among the replicas. Here, green nodes denote the nodes which have identical neighbors in all R replicas ($R = 24$ here); we define these nodes to have a probability $p = 1$. In this example, there are two distinct bubbles with nodes that are, respectively, depicted in this figure by two different colors- i.e., "yellow" and "red" spheres. The configurations inside the bubbles are different for each replica while the tour sections outside the bubbles are the same for all 24 replicas.

Nowadays we have already entered the "Big Data" era. We are generating tons of tons of data everyday where we browse the news on the internet, navigate to a destination by a GPS device, play the songs via mobile app and etc. It is said that the data is the "future oil". Due to the fast-growing computing power and the accumulated data from our daily life artificial intelligence plays a more and more important role in advancing our society. One interesting and important branch from artificial intelligence is called machine learning. Broadly, machine learning is the process of allowing computer programs to parse available data and learn (infer) general rules. The goal of machine learning is to find a model using input data which can be generalized and applied to new data in such a way that model performance increases with increasing amounts of input data. Towards that end, there are two main types of machine learning algorithms: (i) supervised and unsupervised. Unsupervised learning involves training data with unknown labels or associations (ii) Unsupervised learning algorithms seek to label instances based on their connections or commonalities with other instances, via methods such as clustering [8, 9, 10, 11, 12, 13, 14]. On the other hand, supervised machine learning corresponds to learning on training data that has known labels, i.e., data for which the "right answer" is known.

In this dissertation, we will focus specifically on supervised machine learning corresponding to data with either discrete (classification) or continuous (regression) labels. We will introduce our new algorithm that learns by fitting an ensemble of stochastic series expansions to the training data, and then 'votes' on the output of the label. We will demonstrate, through detailed case studies, that this algorithm, which we term the "Stochastic Replica Voting Machines" (SRVM) method, rivals the best performing contemporary models, and additionally surpasses them in various performance metrics. We will demonstrate that the algorithm applies equally well to both classification and regression.

Fig. 1.4 illustrates an example of supervised machine learning class classification. The problem being studied here is called "Four Class" which contains 862 training data points. Each pair of training data contains a vector input which has two dimensions and a binary discrete output(binary class). The goal here is to find out the best

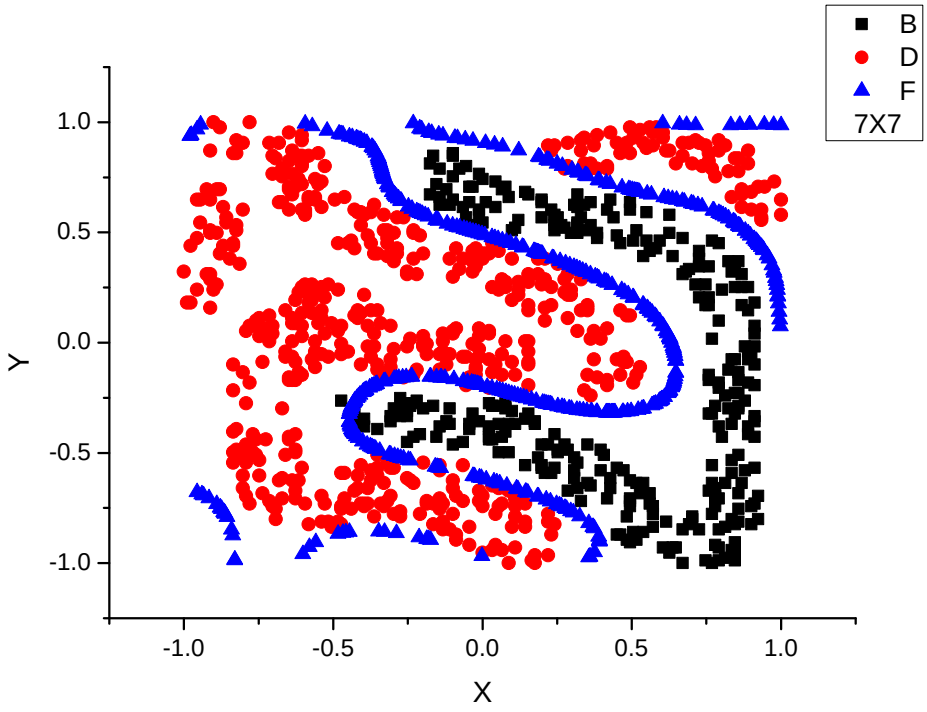


Figure 1.4: (Color online.) B and D represents the +1 and -1 classes respectively in the Four-class data set. The curve F denotes the boundary between these two classes to this Boundary predicted by the SRVM algorithm when using a multinomial of order $n_1 = n_2 = 7$.

boundary that can separate the two output classes well. We are looking for such a boundary that it has the best predicting ability once given another new input data. The above classification process is fulfilled by the SRVM using the Polynomial kernel.

1.5 Prediction on Perovskite and AB solids using SRVM

Recently, there has been a flurry of activity involving the use of Machine Learning, an important subfield of Artificial Intelligence, in the study of materials and complex physical systems, e.g., [1, 3, 5, 6, 2, 7, 8, 9, 10]. Data mining techniques may enable a rapid search through millions of candidate compounds in order to identify promising

technological materials and to potentially predict their detailed properties. Such a task may require far more significant efforts when performed experimentally [12, 11] via the traditional trial and error approach. Machine learning can make such searches far more efficient by systematically pointing to promising materials that may then be fabricated and tested experimentally. In this work, we will focus on two material types: Perovskite and AB solids.

The chemical composition of the Perovskites is ABX_3, where A and B are cations and X is an anion bonding to both cations. In the examples that we will study here, X will be an oxygen anion. Following a standard convention, the A atoms are defined to be the larger of the two cations. A cubic crystal structure is formed by corner sharing BX_6 octahedra as seen in Fig. 1.5.

To ensure stability, the relative size of the A and B cations must, typically, satisfy certain criteria [19]. (Additional illuminating relations between the atomic radii and structure are found in [20].) There are, at least, 95 known stable elements out of which there could be thousands of different possible candidate elemental combinations that may exhibit a Perovskite structure. To experimentally determine the possible properties of these candidate Perovskites would be an arduous if not impossible task. Thus, in recent years, materials scientists turned to *structure-properties algorithms* in order to predict the properties of new theoretical compounds. Along similar lines, in the current work, we will introduce a new algorithm(SRVM) that takes in different elements as inputs and predicts whether or not their combination will result in a stable Perovskite.

The algorithm takes, as an input, the data from 188 different known combinations of A and B ions out of which 29 are capable of forming a Perovskite structure [19]. In machine learning parlance, we are training a new binary classifier over a set of known data. We then apply the trained classifier to investigate hitherto unknown chemical compositions in order to assess their viable formability as stable Perovskite. We will follow the prevalent practice of classifying the stability of candidate Perovskite materials by two figures of merit: (i) the "tolerance ratio", $(r_A + r_X)/\sqrt{2}(r_B + r_X)$ with r_is being the radius of the ions, and (ii) the "Octahedral factor" r_B/r_X.

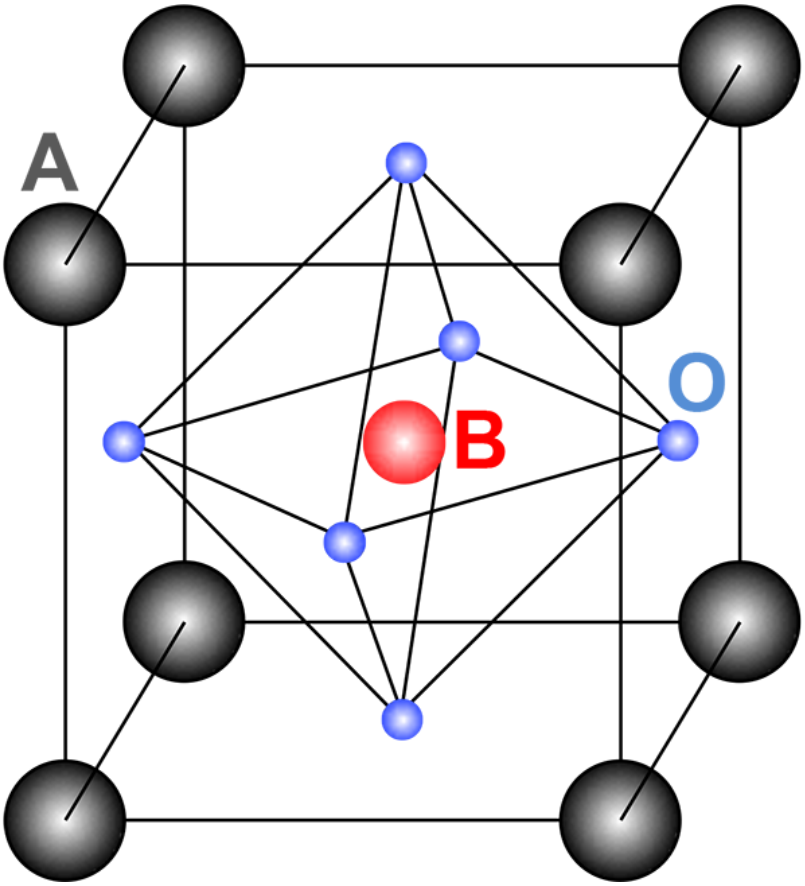


Figure 1.5: Ideal perovskite structure illustrated for ABO_3. Reprinted from reference [18].

Fig. 1.6 provides a summary of our classification results for the Perovskite structure problem for all the algorithms involved in this work. The average accuracy that we reached with the Gaussian kernel from SRVM for determining stable cubic Perovskite oxides was 94.19 %. This accuracy may be contrasted with the performance of the current state of the art SVM package [35]; the SVM method yielded a mean accuracy of 92.53 %.

We next turn our attention, using the data of [36], to the classification of binary solids [26] (of chemical composition AB) into one of $q = 3$ groups (denoted W, Z or R [26, 36]). We employed two commonly used figures of merit as features,

$$\begin{aligned} r_\sigma &\equiv r_s^A + r_p^A - r_s^B - r_p^B, \\ r_\pi &\equiv r_p^A - r_s^A + r_p^B - r_s^B. \end{aligned} \tag{1.1}$$

Here, r_s^A, r_p^A, r_s^B and r_p^B denote the pertinent radii for an electron bound to the A or B ion that is in an s or p orbital.

In Figure 1.7, we provide the phase boundaries (between the W, Z, and R phases) as ascertained by SVM (see the solid curves therein) alongside the boundaries determined by our SRVM method (the domains of the different phases as predicted by SRVM are marked by different colors).

1.6 Overview of dissertation

This dissertation contains information related to the following publications or manuscripts in preparation arranged into chapter divisions as follow: (i) Chapter 1: Z. Nussinov, P. Ronhovde, D. Hu, S. Chakrabarty, B. Sun, N. Mauro, K. Sahu, Inference of Hidden Structures in Complex Physical Systems by Multi-scale Clustering, Volume 225 of the series Springer Series in Materials Science pp 115-138 (ii) Chapter 2: B. Sun, B. Leonard, P. Ronhovde and Z. Nussinov, An interacting replica approach applied to the travelling salesman problem, SAI Computing Conference Proceedings 2016, IEEE Xplore. (iii) Chapter 3: P. Chao, T. Mazaheri, Z. Nussinov, B. Sun, and N. Weingartner, A Stochastic Replica-Based Voting Algorithm For Supervised Learning, to be submitted (iv) Chapter 4: B. Sun, T. Mazaheri, J. Scher-Zagier, D. Magee, P. Ronhovde, T. Lookman, and Z. Nussinov, Stochastic Replica Voting Machine prediction of stable Perovskite and binary alloys, arXiv: 1705.08491v2.

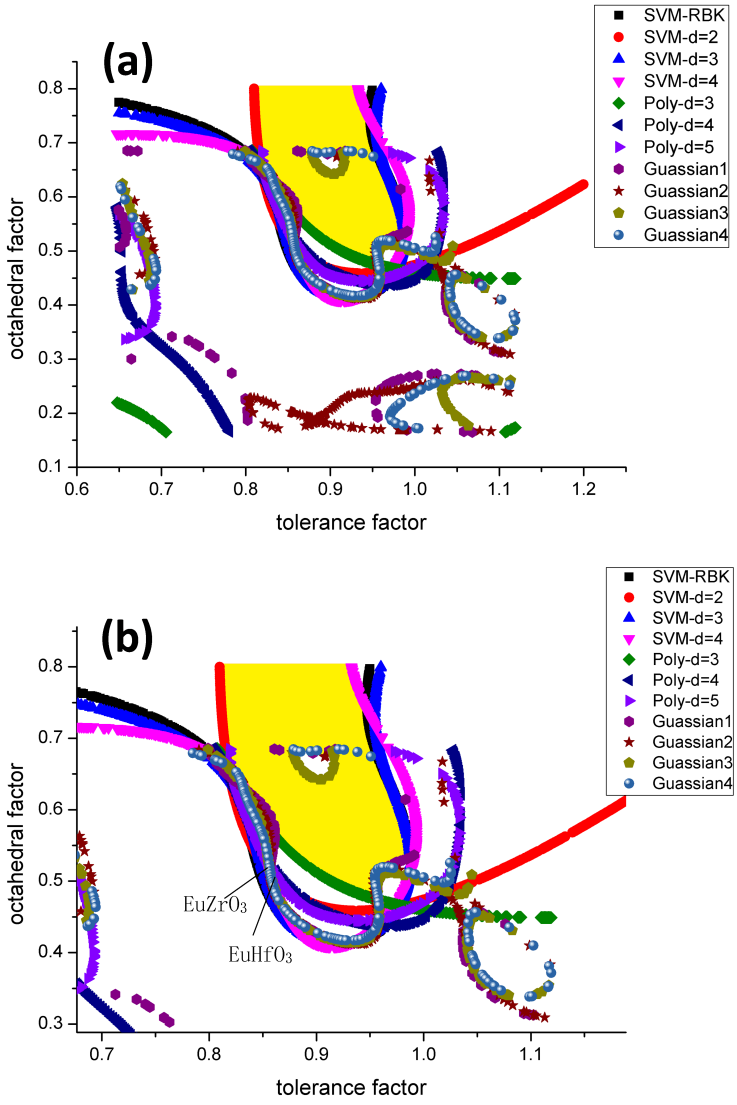


Figure 1.6: The formability of cubic perovskite structure at two different resolutions. The yellow region is that in which all methods (SVM, SRVM with both multinomial and Gaussian kernels) predict that cubic Perovskite structures will form. In panel (a), we show the entire region of measured tolerance and octahedral ratios. Panel (b) provides a zoomed viewed. Two possible candidate compounds that can form cubic Perovskite structure are highlighted: $EuHfO_3$ and $EuZrO_3$.

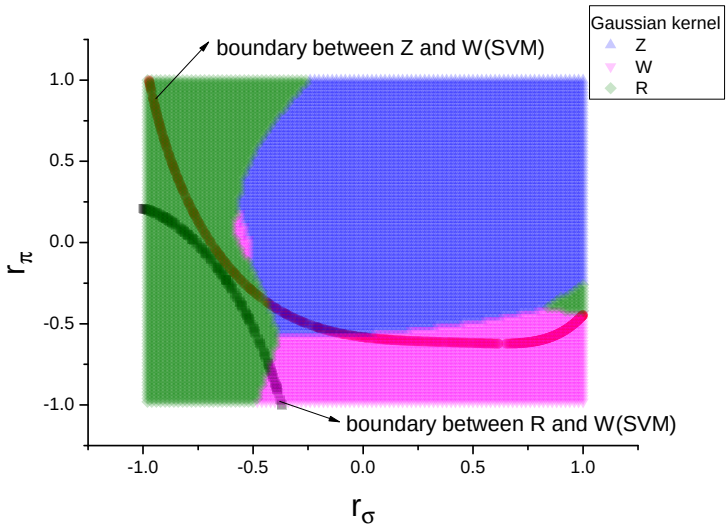


Figure 1.7: Boundary predicted by SVM algorithm and SRVM with Gaussian kernel. The SVM algorithm predicts the boundaries formed by two curves and the SRVM predicts the boundaries formed by the shaded areas. There are three classes of structures related to AB solids that are predicted here. They are called W, Z and R structure respectively.

Bibliography

[1] http://archive.ics.uci.edu/ml/

[2] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, (Courier Dover Publications, 1998).

[3] S. Fortunato, "Community detection in graphs", Physics Reports **486**, 75-174 (2010).

[4] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", Phys. Rev. E **69**, 026113 (2004).

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks", J. Stat. Mech. **10**, 10008 (2008).

[6] Peter Ronhovde and Zohar Nussinov, "An Improved Potts Model Applied to Community Detection", Physical Review E **81**, 046114 (2010).

[7] P. Ronhovde and Z. Nussinov, "Multiresolution community detection for megascale networks by information-based replica correlations", Phys. Rev. E **80**, 016109 (2009).

[8] . V. Gudkov, V. Montelaegre, S. Nussinov, and Z. Nussinov, "Community detection in complex networks by dynamical simplex evolution", Phys. Rev. E **78**, 016113 (2008).

[9] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure", Proc. Natl. Aca. Sci. U.S.A. **105**, 1118-1123 (2008).

[10] T. Mueller, A. G. Kusne, and R. Ramprasad, *Machine Learning in Materials Science, in Reviews in Computational Chemistry*, volume **29**, Edited by A. L. Parrill and K. B. Lipkowitz, John Wiley & Sons, Inc, Hoboken, NJ. doi: 10.1002/9781119148739, chapter 4 (2016).

[11] P. Ronhovde, S. Chakrabarty, M. Sahu, K. F. Kelton, N. A. Mauro, K . K. Sahu, and Z. Nussinov, *Detecting hidden spatial and spatio-temporal structures in glasses and complex physical systems by multiresolution network clustering*, The European Physics Journal E **34**, 105 (2011).

[12] P. Ronhovde, S. Chakrabarty, M. Sahu, K. K. Sahu, K. F. Kelton, N. Mauro, and Z. Nussinov *Detection of hidden structures on all scales in amorphous materials and complex physical systems: basic notions and applications to networks, lattice systems, and glasses*, Scientific Reports **2**, 329 (2012).

[13] J. C. Snyder, M. Rupp, K. Hansen, K. R. Muller, and K. Burke, *Finding density functionals with machine learning*, Physical Review Letters **108** (25), 253002 (2012).

[14] *Information Science for Materials Discovery and Design*, Springer Series Materials, volume **225**, Edited by Turab Lookman, Frank Alexander and Krishna Rajan. 978-3-319-23870-8 (2016).

[15] Yi Zhang and Eun-Ah Kim, *Quantum Loop Topography for Machine Learning*, arXiv:1611.01518 (2016).

[16] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, Nature Physics **13**, 431 (2017).

[17] G. Carleo and M. Troyer, *Solving the quantum many-body problem with artificial neural networks*, Science **355**, 602 (2017).

[18] Maciej Koch-Janusz and Zohar Ringel, *Mutual Information, Neural Networks and the Renormalization Group*, arXiv:1704.06279 (2017).

[19] Arun Mannodi-Kanakkithodi, Ghanshyam Pilania, Tran Doan Huan, Turab Lookman, and Rampi Ramprasad, *Machine Learning Strategy for Accelerated Design of Polymer Dielectrics*, Scientific Reports **6**, 20952 (2016).

[20] G. Pilania, A. Mannodi-Kanakkithodi, B. P. Uberuaga, R. Ramprasad, J. E. Gubernatis and T. Lookman, *Machine learning bandgaps of double perovskites*, Scientific Reports **6**, 19375 (2016).

[21] L. M. Feng, L. Q. Jiang, M. Zhu, H.B. Liu, X. Zhou, and C .H. Li, *Formability of ABO3 cubic perovskites*, Journal of Physics and Chemistry of Solids **69**, 967 (2008).

[22] A. Khan and S. G. Javed, *Predicting regularities in lattice constants of GdFeO3-type perovskites*, Acta Cryst. Section B: Structural Science **64**, no. 1, pp. 120 (2008).

[23] B. Sun, T. Mazaheri, J. Scher-Zagier, D. Magee, P. Ronhovde, T. Lookman, and Z. Nussinov, *Stochastic Replica Voting Machine prediction of stable Perovskite and binary alloys*, arXiv: 1705.08491v2.

[24] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

[25] Yousef Saad, Da Gao, Thanh Ngo, Scotty Bobbitt, James R. Chelikowsky, and Wanda Andreoni, *Data mining for materials: Computational experiments with AB compounds*, Phys. Rev. B **85**, 104104 (2012).

[26] Prasanna V. Balachandran, James Theiler, James M. Rondinelli and Turab Lookman, *Materials Prediction via Classification Learning*, Scientific Reports **5**, 13285 (2015).

Chapter 2

An Interacting Replica Approach Applied to the Traveling Salesman Problem

2.1 Introduction

In this chapter, We present a physics inspired heuristic method for solving combinatorial optimization problems, especially for Traveling Salesman Problem. The concept of a local optimizer acting on some cost function in parameter space is equivalent to modeling a thermal system exploring its energy landscape. At a certain temperature, a thermal system can realistically exchange a certain amount of heat with the environment. While the system is generally attempting to find the lowest energy state, it can temporarily gain energy, and, in so doing escape from a local energy well. However, if the well is deeper than the realistically allowable energy gain, then the system may remain stuck in a metastable, locally optimal energy state indefinitely. This is what happens in spin glasses [2], for instance.

Analogously, if a local well of the cost function in parameter space is deeper than

the realistically allowable positive gain in the cost function, then the simulation of a local optimizer will remain stuck, creating a design tradeoff. The greater potential gain allowed in the cost function, the easier it is for the simulation to escape potential wells, but the longer it will take to actually find a minimum because it will have a larger search space at each step, and it will be moving "uphill" more often. This is the reason that heuristic algorithms can generally be relied upon to produce good pseudo-solutions a few percent above the optimum value, but rarely find the actual global optimum in sufficiently complex problems.

Previous methods such as replica exchange [3] and genetic algorithms [4] have attempted to address this problem with varying degrees of efficacy depending on context. We were inspired to use "information-based replica" correlations and inference to systematically detect ideal subgraph (or "community") partitions of a large graph as two of us have done several years ago [5]. By "replicas" we here allude to independent copies of the same problem. Since then these notions have been applied to a variety of complex system physics (both static and dynamic) and image segmentation problems [6, 7, 8, 9]. More recently, other works applied similar notions to a host of interesting problems [11, 12, 13]. Free-energies and entropies of such ensembles or "multiplexes" have been discussed in [5, 14]. In our approach we do not focus solely on directly extracting the minimum amongst an ensemble of solvers. A key ingredient that we introduced in earlier work is the use of inference to predict which features of the solutions appearing in disparate replicas may coincide with those in the optimal solution; this inference coupling as well as other effective "interactions" between the replicas (e.g., a "geometrical coupling" discussed below) between the replicas may lead to solutions that at intermediate steps elevate the energy (yet lower a "free energy" [5]) similar to the way in which thermal effects may, at intermediate steps, elevate energies in annealing algorithms. Transitions in the complexity of combinatorial optimization problems such community detection problems have crisp signatures in inter-replica correlation functions and information theory measures[10]. Augmenting Refs. [5, 6, 7, 8, 9, 10], we further also note the more recent work of Ref. [15] in which the authors demonstrate that the inference algorithms based on evolving

interactions between replicated solutions in a cavity type approach have better performance in the binary Ising percepton problem. We further note that the "wisdom of the crowds" (which we took in Refs. [5, 6, 7, 8, 9, 10] to be independent replicas) has been long appreciated [16].

The approach that we further develop in the current work- that of geometric interactions between individual solvers and probabilistic ensemble inference- emulates the biological and sociological advantages long known colloquially from collective behaviors and "wisdom of the crowds" [16]. Historically, biologically inspired "swarm intelligence" algorithms [17] have spawned algorithms such as the well known Ant Colony System (ACS) [18] with which we will later compare the new probabilistic variant of our method. In a broad sense, the spin-glass physics cavity approximation inspired message type algorithms of the type of Ref. [15], exchange Monte Carlo [3], genetic algorithms [4], the work that two of us developed in Refs. [5, 6, 7, 8, 9, 10], the ACS, and a multitude of other approaches might all be cast as particular realizations of broad ensemble based interactions or moves.

In the current work, we present new optimization methods based upon the concepts of (i) geometrical distance coupling (GDC) and (ii) inference amongst independent replicas. We apply these tools to the Traveling Salesman Problem. We demonstrate that while a single replica can do quite poorly in solving a challenging problem, via the use of (i) and (ii) above, the *ensemble* of replicas can address much harder problems.

We employ a quantity, which we term the "GDC distance" $C(A, B) \geq 0$ (see Appendix) to measure the similarity between two tours A and B. A distance C(A,B) =0 indicates that tours A and B are identical. We coupled otherwise independent optimizers via their geometrical distances, so that the optimizers will essentially have two "forces" influencing their behavior, see Fig. 1.1. Each optimizer will (i) independently desire to decrease its cost function locally, while simultaneously attempting to (ii) minimize the GDC between itself and all other optimizers (portrayed by the harmonic springs in Fig. 1.1). We demonstrate that through this coupling, local optimizers can escape from wells which otherwise would have confined them permanently,

in the cases we studied.

An additional central tool that we will invoke in this work is that of probabilstic inference from the different replicas, e.g., [5, 8] or a "replica inference based" (RIB) method. In the simplest rendition of this approach, we simply count how many times a given feature of the solution appears in the different replicas. If a structure of the solution (e.g. in the travelling salesman problem that we will discuss in this work, a tour sequentially passing through the same three cities) is common to all or many solvers then one may anticipate this structure appears in the optimal global solution. That is, augmenting the GDC discussed above, the replicas interact effectively with one another via their correlations. By sequentially finding common features in independent copies or replicas of the problem and assuming these to be correct and left untouched, the system to be examined is sequentially made smaller and easier to solve anew. Both of the approaches that we will employ in this work may be viewed as emulating the minimization of an effective multi-replica "free-energy". Schematically, as in Ref. [5], we may consider an effective free-energy given by

$$F = \sum_{i=1}^{R} E_i[\phi_i] - TS[\{\phi_i\}_{i=1}^{R}] \tag{2.1}$$

where ϕ_i are the collection of coordinates that describe solver (replica) i, the quantities $\{E_i\}$ are the energies of contending solutions in the disparate replicas, S is an inter-replica correlation functional, and $T > 0$ sets a relative weight between the sum of the intra-replica contributions ($\{E_i\}$) and the inter-replica correlations (that may, e.g., imitate the GDC, RIB or other couplings). The detailed iterative minimization procedures that we describe in this work for the Traveling Salesman Problem are particular simple examples of the more general idea embodied by the minimization of the replica ensemble functional of Eq. (2.1). The springs in Fig. 1.1 symbolize inter-replica effects. For finite T, both intra-replica and inter-replica effects must be assuaged when minimizing F.

2.2 Algorithms for the Traveling Salesman Problem

There are roughly three classes of algorithms for Traveling Salesman Problem. The first class is the greedy heuristic which gradually forms a tour by adding a new city at each step, such as the *Nearest-Neighbor algorithm* [21]. In our method, we use the Nearest-Neighbor algorithm to initialize a candidate tour construction. Briefly, the Nearest-Neighbor algorithm is given by three steps: (1) Select a random city. (2) Find the nearest unvisited city and go there. (3) Check if any unvisited cities are left? If yes, go to step 2. If no, return to the first city.

The second class of heuristic TSP algorithms is a tour improvement approach. A typical example is given by the *k-opt algorithm* which seeks to iteratively improve the current state of a tour by removing k edges and replacing them in the most optimal way by random searching on a limited number of cities at a time. A famous local search algorithm by Lin-Kernighan (LK) [22] belongs to this class. The LK algorithm is a variable k-opt algorithm which decides which k is the most suitable at each iteration step. This makes the algorithm quite complex, and few have been able improve it. A more in-depth study of the LK algorithm with possible improvements was made by Helsgaun (LKH) [23].

The third class of heuristic methods is a composite algorithm that combines the features of the former two. A good example can be found in Dorigo and Gambardella [24] where the authors combined the ant colony system (ACS) [18] with the 3-opt method to achieve strong results. Here the ACS acts like a more sophisticated tour construction algorithm which allows communication (pheromones left by individual ants) between different ant solvers. 3-opt is the local optimizer which helps to optimize the results obtained with ACS.

2.3 Geometrical distance coupling algorithm

As noted above, we use the geometric distance coupling between different solvers (or replicas) to enhance the solutions found by individual solvers. To demonstrate the strength of our approach and the degree to which coupling between replicas can dramatically improve the results, we apply our method on the bare k-opt algorithm. On their own, sans the use of replicas, the k=2 or 3 opt-algorithms have a very poor performance; this makes the improvement using our replica based approach very clear. Our GDC replica based approach may, in principle, be applied to any algorithm (not solely k-opt). The GDC algorithm is implemented as follows:

1. Use the Nearest-Neighbor Algorithm [21] to seed all the replicas, beginning at random cities in different replicas to ensure some variation in the initial states.
2. Perform a variable initial number of k-opt steps independently on all replicas.
3. Apply the GDC step (after a given number of iterations) as follows:
   (a) Determine the most common edge among all replicas.
   (b) For replicas that share the identified common edge (see Fig. 1.2 as well as Appendix for further discussion), attempt to move a random city to its average position relative to common edge in all relevant replicas. Allow the move only if the total tour length is decreased or if it is increased by less than a specified tolerance.
4. Perform a variable number of k-opt steps independently on all replicas.
5. Go to step 3 until a global number of iterations is reached.

To clarify, we start the Nearest Neighbor algorithm in different cities for each replica and perform an initial number of k-opt steps in order to guarantee that our replicas are not starting too close to each other in parameter space. The GDC attempts to ensure that all of the replicas eventually converge on the same location in

parameter space. If the replicas become clustered too closely together (as in, e.g., the cartoon of Fig. 1.1), they may not efficiently explore the landscape of possible solutions and fail to find the global minimum.

During the GDC step, we randomly select a sequence of cities. We then calculate the average position of those cities relative to the most common edge for the replicas which contain the identified common edge. Then, for each relevant replica, that city is plucked from its current location and placed in the average position. The locally broken tour sequence is reattached in the manner described in Appendix. If the tour length is either decreased or increased by less than the tolerance, then the move is accepted, otherwise it is rejected. In this manner, the algorithm is able to locally decrease the quantity

$$Q_j \equiv \frac{1}{r} \sum_{i=1}^{r} |S_{i_j} - \overline{S}_j|. \tag{2.2}$$

In the above, r is the number of the relevant replicas which share the common edge in step 3 of GDC algorithm, $1 \leq j \leq N$ is the city index, S_{i_j} is the distance between city j and a common edge in replica i, and $\overline{S}_j$ is the average of S_{i_j} (as averaged over all the relevant r replicas). By lowering Q_j for different cities j, the replicas generally become more similar to one another.

2.4 Results from geometrical distance coupling algorithm

We tested our algorithm on several instances from TSPLIB [25] using $k = 2$ and 3 for the k-opt step. We demonstrated that within a certain range of N for which 2-opt and 3-opt alone almost always failed to find the global minimum, our enhanced algorithm was able to find it in a reasonable amount of time. For some larger values of N, our algorithm also failed to find the minimum, but it did significantly improve the k-opt estimate, and we believe that it could find the minimum if mated with an

appropriate optimizer which is more sophisticated than the standard 2- and 3-opt methods.

The results are summarized in Table 2.1 where length and time values are averaged over 10 runs. The unit of the CPU time here is second. The GDC enhanced algorithm is labeled "2-opt GDC" and "3-opt GDC" respectively, for the base 2- or 3-opt local search routine. The parameters used here were $R = 20$ replicas, 1 geometrical distance coupling step consisting of $N - 2$ moves alternated with 1000 to 15 000 k-opt steps, and an allowable increase in the tour length of 0.2% - 1% during each attempted GDC move. For all instances studied in Table 2.1, we performed 1 000 000 initial k-opt steps independently on all replicas for step 2 in Sec. 2.3. For the instances with a small number of cities (berlin52, eil51, pr76, eil76) we used approximately 1000 k-opt steps in step 4. For the instances with a relatively large number of cities (ch130, ts225, a280, lin 318, and att532), we invoked 15 000 k-opt moves in step 4 of the algorithm.

We allowed larger tour length increase tolerance in step 3(b) for larger N problems. The GDC method using the 3-opt optimization can correctly solve all examined TSPLIB [25] problems up to 280 cities. If 2-opt is applied instead of 3-opt the maximal solvable size is 225. We note that neither the bare 2- nor 3-opt by themselves are not able to find the optimal solution for even the smallest of these examples given a comparable number k-opt optimization steps. The time required to find the global optimum with the GDC step is large compared to the computing time for the k-opt. Part of the reason is that the k-opt optimization is easily trapped in local minima, but the GDC step is capable of pulling the optimizer from the local minima and having them explore a much broader region of the solution space.

We investigated the effect of the number of replicas used on the performance of the algorithm. Figure 2.1 and 2.2 contrast results obtained when our GDC algorithm is applied with $R = 5$ and $R = 20$ replicas to improve the bare 2-opt and 3-opt respectively (we term the resultant algorithms 2-opt GDC and 3-opt GDC) in the lin318 problem. The average tour length in the $R = 5$ case using the 2-opt GDC was 42 479 whereas when using $R = 20$ replicas, the 2-opt GDC yielded a path of

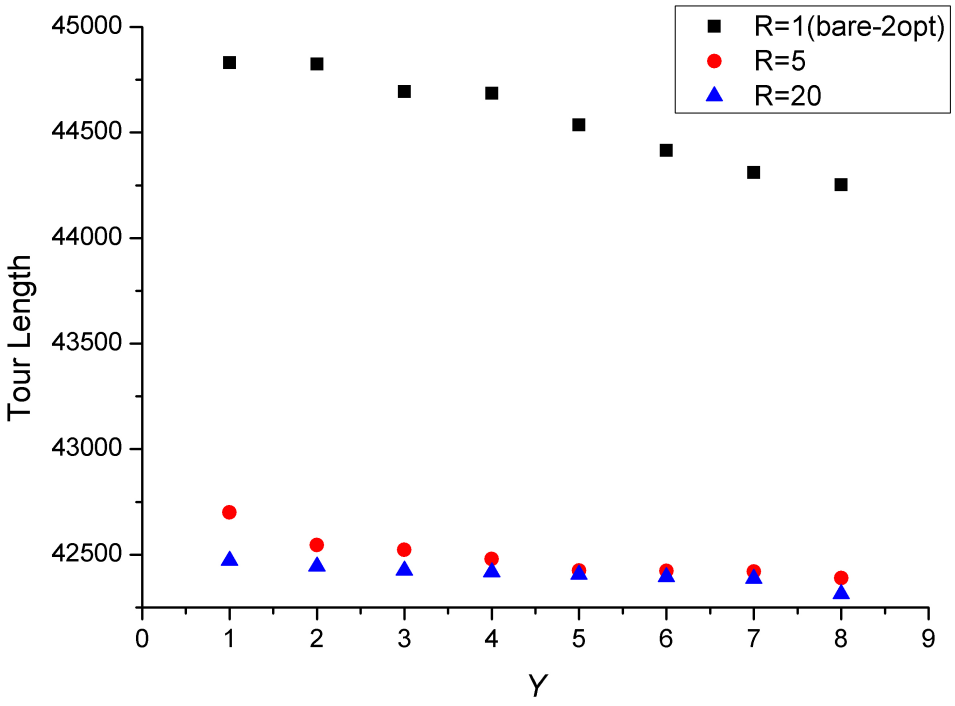


Figure 2.1: (Color online.) The improvement of the bare 2-opt method by the use of replica coupling for the lin318 problem. The figure shows that tour lengths found by invoking R=1 (black squares), R=5 (red circles) and R=20 (blue triangle) replicas averaged over $Y \leq 8$ solution attempts. The horizontal axis shows the results obtained by including Y attempts. Applied to the 2-opt GDC, the use of $R = 20$ replicas produced better results than the use of $R = 5$ replicas.

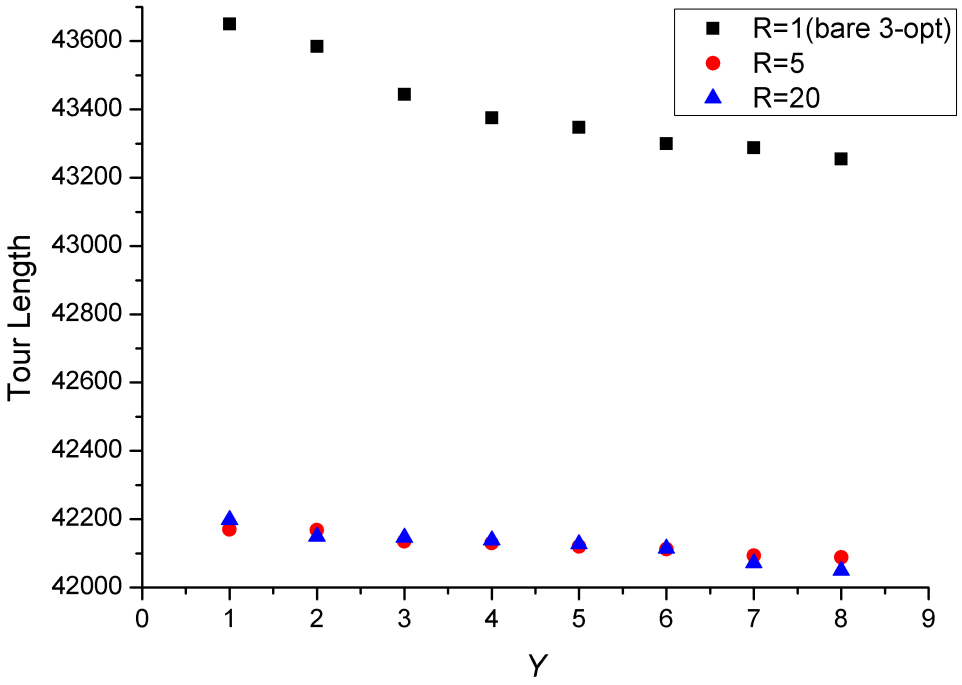


Figure 2.2: (Color online.) The improvement of the bare 3-opt method by the use of replica coupling for the lin318 problem. For 3-opt GDC the average tour length for the tested range of replicas was very close, but when $R = 20$, the algorithm still found a smaller tour length.

distance 42 404. Not too surprisingly, in both the 2-opt GDC and the 3-opt GDC, the $R = 20$ replica case provides shorter tour lengths than the $R = 5$ replica algorithm; this improvement with increasing R is smaller for the 3-opt GDC (as the bare 3-opt algorithm is better than the 2-opt method).

2.5 Probabilistic replica-inference based algorithm

Next, we show how we developed a replica-inference based (RIB) algorithm to, e.g., solve the lin318 test problem and the att532 test problem. The RIB algorithm is implemented as follows:

1. Use GDC Algorithm to seed all the replicas (the total number of replicas is R).

2. For all the replicas, calculate the probability distribution of the nodes (see equation (2)).

3. Given the probability information of the nodes, divide the tour into different parts: "bubble" region in which different tours appear in disparate replicas and a "backbone" region which is common to all replicas (see Figs. (2.9, 2.13)).

4. Keeping the common backbone region unchanged, examine different tours in the bubble regions such that when combined with the backbone path, they will lead to new viable solutions (replicas) and then pick the one (replica X) that has the shortest path. When we examine different configurations in the bubble regions we must pay attention to the "pairing" inside the bubbles (see Figs.(2.9,2.13) and discussion below). We also observe that inside a given bubble there are no lines that cross as required by the triangle inequality (see Fig. 2.4).

5. Two replica comparison (performed R times): compare the replica X found in step 4 with the R original replicas that existed prior to step 4 through steps 2-4.Each time after step 4, update the replica X found in step 4. A final solution will be produced after R times of two replica comparison.

In the up and coming, we will introduce and invoke a probability p_j associated with each node j. This probability will measure the frequency that the links to the incoming (i) and outgoing (k) associated with each city j are the same amongst all replicas. That is,

$$p_j = \frac{M_j}{R}. \tag{2.3}$$

M_j is the number of times that the composite link $\langle ijk \rangle$ connecting the three cities i, j and k appears in the ensemble of R replicas studied.

In what follows, we introduce the concept of a "*bubble*" alluded to in the algorithm above. A "bubble" is, by fiat, comprised of all nodes j for which the composite links $\langle i, j, k \rangle$ are not the same across all replicas (i.e., nodes for which $p_j < 1$). The set of such nodes must generally terminate somewhere and is linked to a backbone of nodes that have the same links in all replicas. The termination points marks the boundaries of the "bubbles". In the more detailed representations of the tour solutions in some of the figures that follow, we will typically mark green all points j for which the links $\langle ijk \rangle$ are identical in all replicas (i.e., the associated probability $p_j = 1$).

In Fig. 2.5, we schematically depict typical "one- in-one-out" and "two-in-two-out" bubbles (i.e., bubbles that are attached to the common backbone by either two or four points).

2.6 Results of the probabilistic replica inference approach

We next apply our replica-inference based algorithm to solve the lin318 test problem from TSPLIB (see 3-opt GDC results in Table 2.1). Typically, we used $R = 24$ replicas. The known true (i.e., minimal distance) tour solution is a path of length 42 029. Each of the $R = 24$ replicas employed provided a contending solution; the paths in each of the replicas that varied in length from 42 050 to 42 199.

A significant fraction of the $R = 24$ configurations produced by the GDC algorithm (discussed in sections 2.3, 2.4) when it is applied for each of the R replicas share three-site configurations of the type shown in Fig. 2.3. Schematically, the composite link $\langle ijk \rangle$ may be shared amongst numerous replicas as is illustrated in Fig. 2.6. To quantitively measure this similarity, we employ (as we have alluded to in sections 2.3) a probability distribution p_j based on the these link patterns $\langle ijk \rangle$ associated with each node j. That is, in every replica each node has two adjacent nodes, so p_j is defined as the max number of replicas for which j shares the same neighbors (where j is in the middle) divided by total number (R) of replicas ($R = 24$ in our example calculation here). In Fig. 2.7, we show a representative plot the probability distribution for different nodes on the lin318 problem.

We wish to take advantage of the information contained in all of replica tours. Toward that end, we observe in Fig. 2.7 that 162 out of 318 nodes (approximately 50% of the nodes) have a probability of 1. We conjectured that the common structures for nodes which have a probability equal to 1 are the same as that from the known optimal solution. To test whether it was the case, we made a plot of Fig. 2.8. In Fig. 2.8, p_j is a given fixed probability p_j defined above when averaged over all replicas. Then we looked at the set of all links $\langle i, j, k \rangle$ having that probability p_j. q is the fraction of these links that appear in the optimal solution; we then plot q versus p_j in Fig. 2.8.

Perusing Fig. 2.8 (associated with the lin318 problem), we observe that links $\langle i, j, k \rangle$ with inter-replica frequency $p = 1$ (i.e., links that consistently appeared in all replicas) indeed appeared in the optimal tour. Furthermore, numerous links $\langle i, j, k \rangle$ with $p < 1$ (i.e., those which were not consistent across all replicas) also appeared in the optimal shortest tour solution. In what follows, we introduce the concept of a "bubble" as it pertains to the current problem. A "bubble" is, by fiat, comprised of all nodes j for which the links $\langle i, j, k \rangle$ are not the same across all replicas (i.e., nodes for which $p < 1$). The set of such nodes must generally terminate somewhere and is linked to a backbone of nodes that have the same links in all replicas. The termination points marks the boundaries of the "bubbles".

In the replica-based inference approach, the common structures with $p = 1$ are left untouched. We aim to solve the smaller and less difficult bubble problems separately instead of the entire tour map. In Figs. 1.3 and 2.10, nodes (represented as spheres in a connected tour map) whose probability is $p = 1$ are colored green and those with $p < 1$ are colored red or yellow.

We now turn to step 3 of our algorithm. By definition, the bubbles encompass the same set of nodes in all replicas. That is, if there is at least one replica in which a red (or yellow) node a is attached to another red (or yellow) node b, then a and b will lie in the same bubble for all replicas. In the lin318 problem, we obtained two bubbles by comparing the 24 replicas to each other. This is illustrated in Fig. 1.3. (Different colors refer to different bubbles.)

Next, we apply step 4. As mentioned previously, the green nodes in Fig. 1.3 remain unchanged. We then solved for the optimal tour inside the two identified bubbles for this problem. The shortest intra-bubble tour for the larger bubble (red nodes) is 26 591 (from replica 7). The shortest intra-bubble tour for the smaller bubble (yellow nodes) is 578 (this also appeared in replica 7). Although we cannot find the optimal solution for lin318 problem at this stage, the current best tour length is 42 050.

Caution must be exercised in choosing the optimal "intra-bubble" tour among all the relevant replicas. We mandate that the resultant tour is a valid TSP solution (i.e., we need to make certain that (i) the tour visits each node exactly once inside any bubble and that (ii) the formed global tour forms a closed path). To that end, we should consider the *Pairing* between two nodes alludes here to the circumstance that these two nodes are continuously connected to each other by an intra-bubble segment (see Figs. (2.9, 2.13)). Figs. (2.9, 2.13) constitute top views of the tours depicted in Figs. (1.3, 2.12) where the common backbones and regions with differing node paths ("bubble") are made clear. The blobs in Fig. 2.9 schematically denote bubbles. The green solid lines (backbones) are the tour path formed only by common structures (which we do not want to change). The dotted lines inside the blobs are possible tour paths (we want to find the shortest such paths). Within the blobs there might be some common structures between the different replicas. Among all of the 24 replicas

Table 2.1: TSP performance and results using 2-opt, 3-opt, 2-opt GDC, and 3-opt GDC on a selection of TSPLIB instances. k-opt GDC are results from the current work. The values of tour lengths and CPU times are averaged over 10 runs. For the studied instances, 2-opt and 3-opt always failed to find the global minimum alone. With geometrical distance coupling (see Sec. 2.3), our 2-opt GDC algorithm found the global minimum up to $N = 225$ cities, and 3-opt GDC found the optimal tour up to $N = 280$. Although neither 2-opt GDC nor 3-opt GDC found the optimal solution for lin318, they significantly improved the base 2- or 3-opt estimate. The percentages above the optimum for lin318 was 5.8% for 2-opt and 3.1% for 3-opt which was reduced to 0.94% and 0.22% for 2-opt GDC and 3-opt GDC, respectively.

Benchmark problems			Bare $k-opt$ algorithms				Results from our new method			
			2-opt		3-opt		2-opt GDC		3-opt GDC	
Problem	Cities	Optimal length	Length	CPU time	Length	CPU time	Length	CPU time	Length	CPU time
berlin52	52	7542	7721	0.035	7606	4.3	7542	2.31	7542	1.62
eil51	51	426	433	0.023	429	13.41	426	6.82	426	39.36
pr76	76	108159	110875	0.057	109096	27.32	108280	8.511	108159	399.57
eil76	76	538	553	0.05	544	29.35	538	50.3	538	184.63
ch130	130	6110	6354	5.09	6232	8.377	6110	1003	6110	410
ts225	225	126643	128103	7.08	126885	110.4	126643	3398.6	126643	76.44
a280	280	2579	2701	8.88	2642	143.43	2615	1500	2579	7807.8
lin318	318	42029	44473	7.66	43347	48.57	42423	15120	42124	28080
att532	532	27686	29338	10.88	28447	60.00	28607	20220	28035	40770

there were the two possible ways of pairing for the bubble nodes at the boundary. The bubbles in Fig. 2.9 are of "two-in-two-out" type. That is, the full tour will enter and exit each bubble twice. As discussed above, when we pick the optimal solution for each bubble and combine them together to form a new global solution we must make it certain that the tour is still valid.

Step 5: We were able to decrease the old tour length for replica 7 by using just two replicas. In doing so, we find that we can decrease the tour length from 42 050 to 42 029 by swapping common bubble appearing in both replicas 3 and 7 and having the same "in-out" pairing. The bubbles being swapped are of the "two-in-two-out" type with the same pairing (see Fig. 2.9). So we can safely swap these two bubbles. The results are shown in Figs. 2.10 and 2.11. The total tour length associated with the common bubble in replica 3 is 4042 compared to 4063 for replica 7 with a difference of 21. Upon swapping the lower distance tour in the smaller bubble from replica 3 with the existing one in replica 7, replica 7 attained the ideal optimal tour length

of length 42 029- the correct solution of the lin318 problem. The resulting tour is depicted in Fig. 2.11.

We also applied our replica inference method to further optimize the solutions obtained by the GDC algorithm for the att532 problem [25]. In Fig. 2.12, all 24 replicas were employed to produce the common backbone ("green") nodes and bubbles (differing tour regions attached to the common backbone) as we did for the lin318 instance. There were six bubbles in total. Five of these bubbles were of the "one-in-one-out" type while the rest were of the "three-in-three-out" type (see Fig. 2.13). In all of the replicas, the "in-out" pairing was between the very same sets of node pairs. For the five smaller bubbles, by virtue of their minute size, brute force minimization quickly produced to find the optimal solution. For the "three-in-three-out" bubble we inserted the shortest path result (that obtained from replica 17) among the 24 replicas. These steps led to a solution for the att532 problem having a total tour length of 27 937 as compared to the average replica result of 28 035. We tried to decrease the tour length of the big bubble of "three-in-three-out" type further by invoking replica pair comparisons. This iteratively led to a replacement of the original three-in-three-out bubble to many far smaller bubbles. We then investigated whether we can optimize the original big bubble by swapping these smaller bubbles between replica 17 and others. Adopting some smaller bubble solutions from replicas 1,10, and 23 respectively, this set of sequential minimization and inference operations led to a better solution having a tour length 27 881. The process is illustrated in Fig. 2.14. The length of the resultant contending solution is 27 881. (The known optimal shortest path for att532 has a tour length equal to 27 686.)

Although we still cannot find the global minimum for att532, the solution of 27 881 produced by GDC algorithm and bubble method togother is only 0.7% above the optimum. More importantly, by employing inference, we were able to reduce the large problem involving a minimization of the path of all nodes in the graph to a set of smaller problems involving disparate "bubbles". Other than the "three-in-three-out" bubble the rest tour configuration was found to be the same as the known optimal solution.

To better understand the difference between our solution and the known optimal solution, we compared our replicas to the known optimal solution. Perusing Fig. 2.13, we find that despite of the same in-out "pairing" occuring associated with identifying the locations where the tour went in and out of each bubble, the optimal solution tries to go from node (on boundary) A6 to A5 more directly while our replicas always intend to go north from city A6 and finally reach A5 after visiting numerous nodes.

2.7 Conclusion

We introduce *a general method for improving known algorithms.* (i.e., the 2-opt and 3-opt of the TSP [21]). Although, for definitiveness, we focused on the TSP, the premise of underlying our method is very general and it may, in principle, be applied to many other problems. The core concept of our approach is that of *coupling between independent solvers* (see Fig. 1.1 and Eq. (2.1)). Such a coupling between members of an *ensemble* of solvers (or "replicas") that collectively seek to find an optimal solution may substantially improve the convergence to the correct answer as compared to the prevalent single replica algorithm. This coupling may be introduced amongst solvers of many types (with these solvers obtained by any previously known algorithm). We underscore and reiterte that by couplings these solvers, we may, very significantly, improve earlier results. In the context of the TSP, we demonstrated that geometrical distance and probabilistic inference coupling between otherwise independent replica solvers allow local solvers to flee from false local minima. By doing this, we obtained optimal TSP tours even when single solvers were unable to find the correct answer. As an example, we showed that while the bare 3-opt method fails to solve for the examined TSP problems with more than 50 cities (see Table 2.1), *by invoking replicas, the 3-opt method can accurately solve problems up to size of 318 cities* (see Fig. 2.10). We furthermore obtained nearly optimal solutions even for larger systems. For instance, in the att532 example the replica method led to a solution with 0.7% increase in tour length relative to the

minimum (see latter part discussion of Section 2.6). Thus, the inter-replica coupling indeed led to a substantial improvement. Unlike genetic algorithms that involve no such coupling and simply randomly swap the city nodes among different contending solutions, our algorithms makes use of replica correlations and inference. Genetic algorithms fail to solve problems as complicated as those we do. To our knowledge, the currently best genetic algorithm [26] already falters in atempting to correctly find the minimal path for a 76 city tour ("pr76") that we readily solved here (see Table 2.1) and successfully went to far larger city tours. In conclusion, we introduce a new replica based approach that may be applied to disparate problems beyond the confines of the particular TSP problem solved here and the clustering and image segmentation challenges addressed in [5, 8]. Our core idea is that even algorithms that are simple may be much more potent once *inter-replica interactions* and *inference* are invoked.

2.8 Appendix

The geometrical distance coupling step is applied "on top of" a base TSP solver. Generally speaking, the idea is to alternate optimization of the local solver (k-opt in the current work) with the GDC step to induce the algorithm to escape local minima and enhance the chances of finding the globally optimal tour solution. GDC seeks to utilize the distance information implicitly contained in multiple TSP solvers to enhance the optimization performed by a base algorithm.

For illustration purposes, the following discussion uses only five replicas which are labeled a, b, c, d, and e. We then represent a candidate tour solution as a string of N cities,

Replica a: tour: $a_1, a_2, a_3, \ldots, a_N$

Replica b: tour: $b_1, b_2, b_3, \ldots, b_N$

Replica c: tour: $c_1, c_2, c_3, \ldots, c_N$

Replica d: tour: $d_1, d_2, d_3, \ldots, d_N$

Replica e: tour: $e_1, e_2, e_3, \ldots, e_N$

where each replica tour correspondes to a permutation of the integers $(1, 2, 3, 4, \ldots, N)$. The GDC algorithm is given by the following steps:

1. Determine the most common edge among all replicas:

   (a) Randomly select a "standard" reference city from $2, 3, 4, 5, \ldots, N-1$. Cycle each replica order so it includes the standard city as the first element. The five example replicas have the following configurations:

   Replica a: $S, a_2, a_3, \ldots, a_N$

   Replica b: $S, b_2, b_3, \ldots, b_N$

   Replica c: $S, c_2, c_3, \ldots, c_N$

   Replica d: $S, d_2, d_3, \ldots, d_N$

   Replica d: $S, e_2, e_3, \ldots, e_N$

   where a_2, b_2, c_2, d_2, e_2 are the neighbors of S in the corresponding replica.

   (b) Determine the most common edge among all replicas. That is, find which of the five links ($S - a_2$, $S - b_2$, $S - c_2$, $S - d_2$, $S - e_2$) appears the most frequently and flag the corresponding replicas. Let's call the most common nearest neighbor city "S'". Suppose there are three replicas containing this link $S - S'$:

   Replica a: $S, S', a_3, \ldots, a_N$

   Replica b: $S, S', b_3, \ldots, b_N$

   Replica c: $S, S', c_3, \ldots, c_N$

   (c) Calculate the average position for every N cities. For example, replica a has a long link: $S - S' - a_3 - a_4 - \cdots - a_{N-1} - a_N$. We can know the distance from a specific city (say a_4) to the first city S. For the relevant replicas (replicas a, b, and c in this example) calculate the average value of the distance of all cities to the standard city S.

2. For replicas that share an identified common edge, attempt to move a random city to its average position (measured relative to the common edge). For example, in replicas a, b, and c, we already know the average distance of city a_3 to

S. We compare this value to the distance of all N cities to the standard city, and we find that the distance of a_5 to S is the closest to the average distance of city a_3 to S. If the total tour length after this change is decreased or if it is increased by less than a specified tolerance, we rearrange the order as follows: $S, S', a_4, a_5, a_3, a_6, a_7, \ldots, a_N$. Similarly, we continue to move random cities to their average positions in all relevant replicas for N-2 times.

The geometrical distance coupling $C(A, B)$ between two replicas (candidate tours which share at least one common edge) is calculated by

$$C(A, B) = \frac{1}{N} \sum_{i=1}^{N} D_i \tag{2.4}$$

where N is the number of cities in the instance. D_i represents the difference of the geometrical distance from the i^{th} city to the standard city in tour A and tour B.

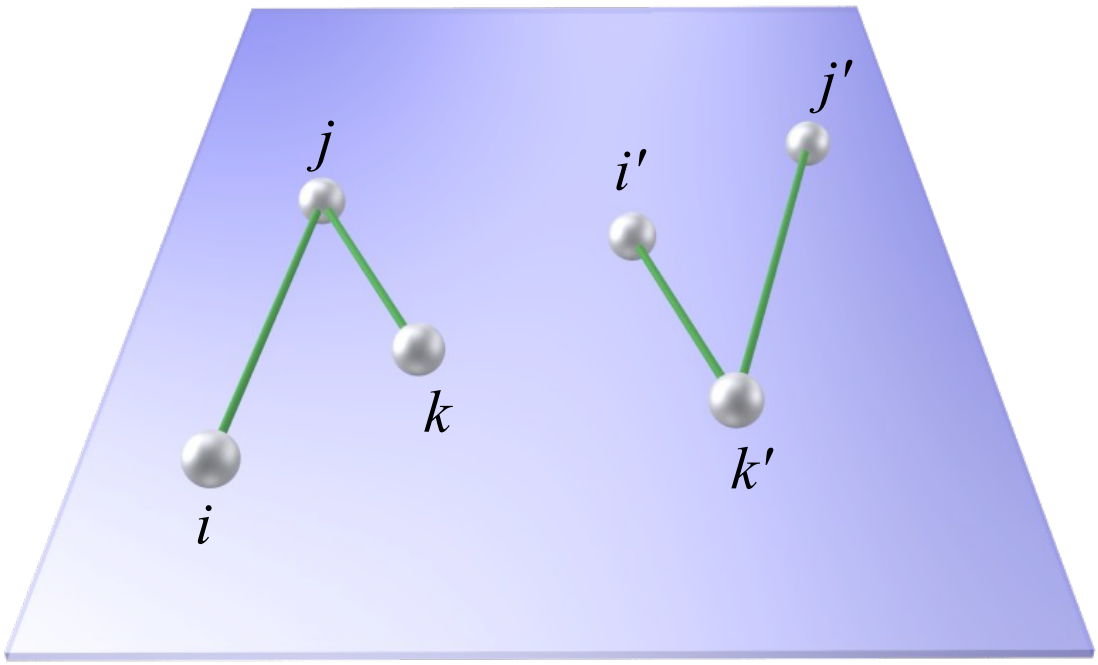


Figure 2.3: (Color online.) A schematic representation common structure segments in solutions of the Traveling Salesman Problem. Cities in segments i, j, and k as well as i', j', and k' are represented by spheres and the solved tour path follows the depicted edges connecting the cities.

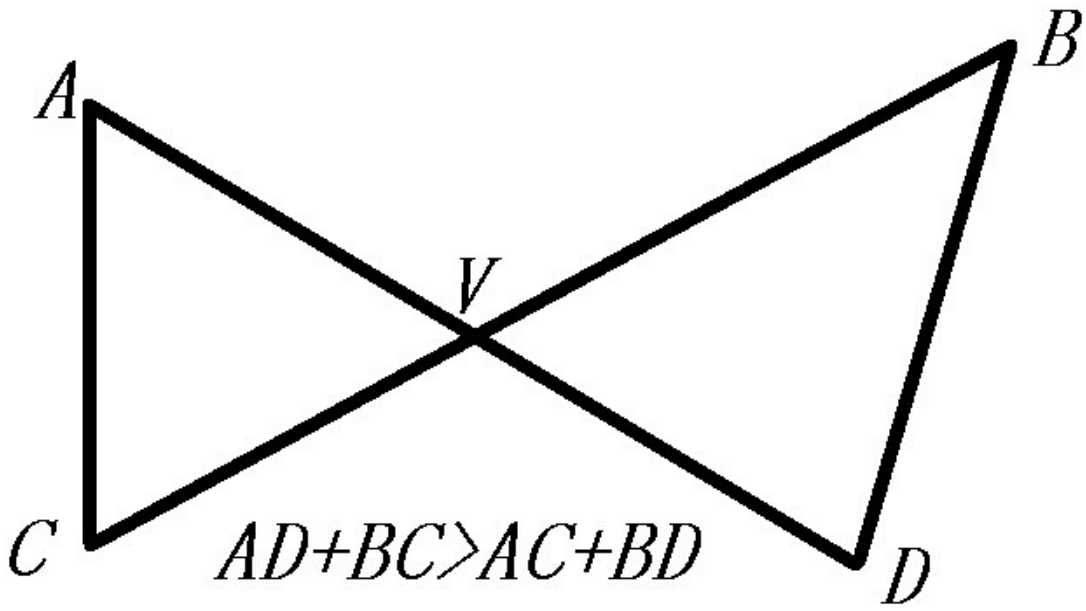


Figure 2.4: (Color online.) A cartoon illustrating that the minimal tour will never intersect itself. In the figure above, a tour containing the two segments AC and BD will always have a shorter length than a tour incorporating the same four points yet includes the segments AD and BC (that intersect at a point V). The proof of this assertion is trivial. By the triangle inequality as applied to the triangles ΔAVC and ΔBVD respectively, we have $AV + VC > AC$ and $BV + VD > BD$. Adding these two inequalities yields $AD + BC > AC + BD$. Permuting the contour segments (e.g., AD,BC $\rightarrow$ AC, BD) to avoid crossing will always lower the total path length.

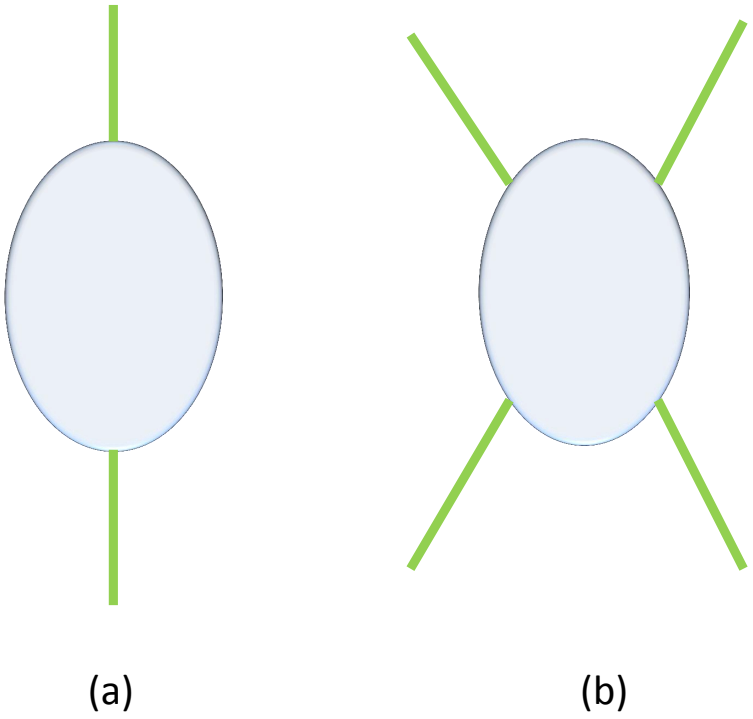


Figure 2.5: (Color online.) Typical “one-in-one-out” and “two-in-two-out” bubble.

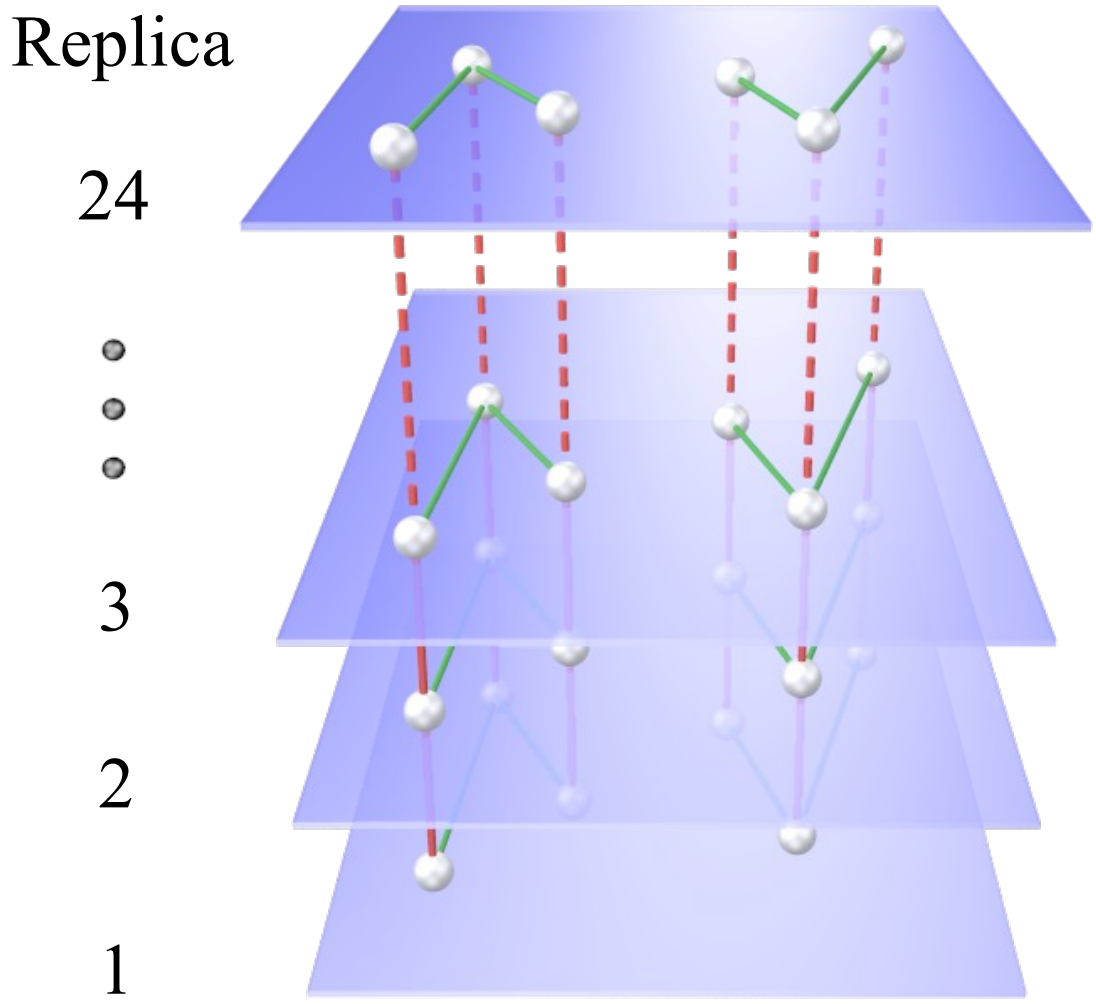


Figure 2.6: (Color online.) Schematic representation of common structures that appear in various replicas. In this example, the candidate common structure contains of three nodes with two links between them. When the common structure appears in *all* replicas exactly, we define the probability to be $p_j = 1$ ($p_j = 24/24 = 1$ here). When the structure does not appear the same in all replicas, $p_j < 1$. For example, if $p_j = 1/3$, the number of replicas that contain the common structure is eight ($p_j = 8/24 = 1/3$).

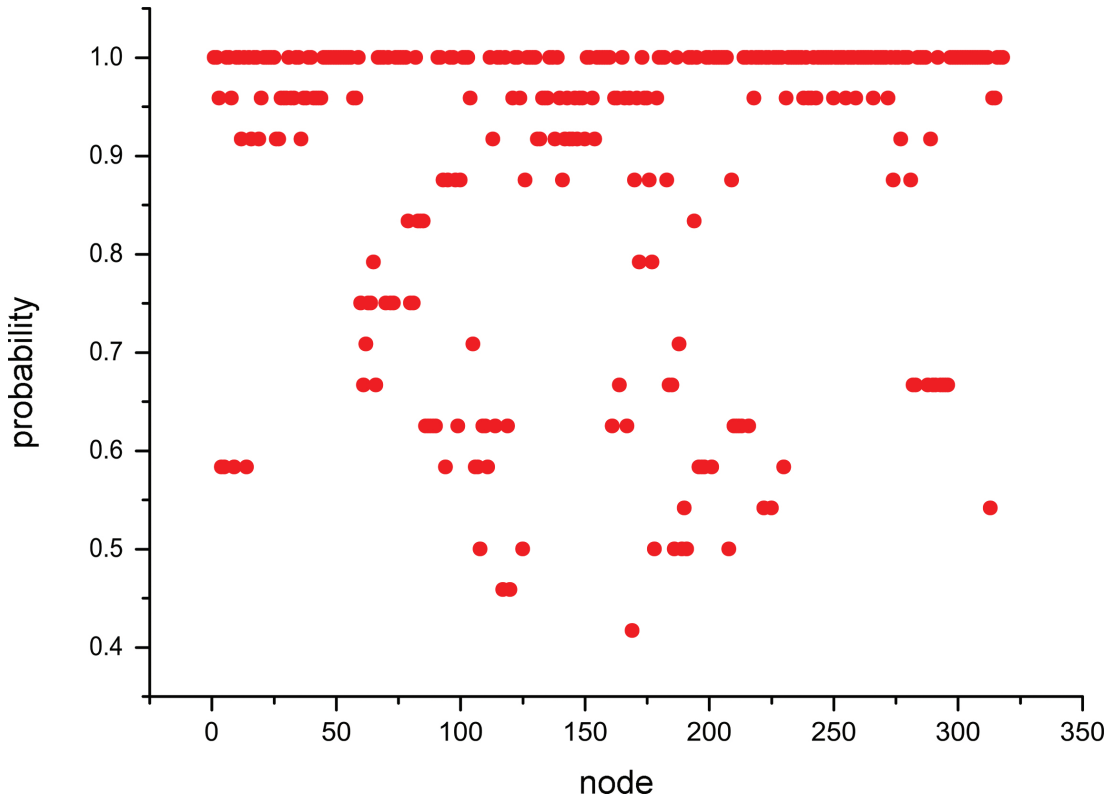


Figure 2.7: (Color online.) Building on the abstract representation in Fig. 2.6, we plot the exact probability distribution p_j (frequency of common structures identified in each of the different replicas) for each node in the lin318 problem from TSPLIB. Here, there are $N = 318$ nodes and $R = 24$ replicas. Every node has two adjacent neighbors, so we define p_j as the number of times a common structure occurs (, the same pair of neighbor cities are connected to node j) among the replicas divided by the total number of replicas.

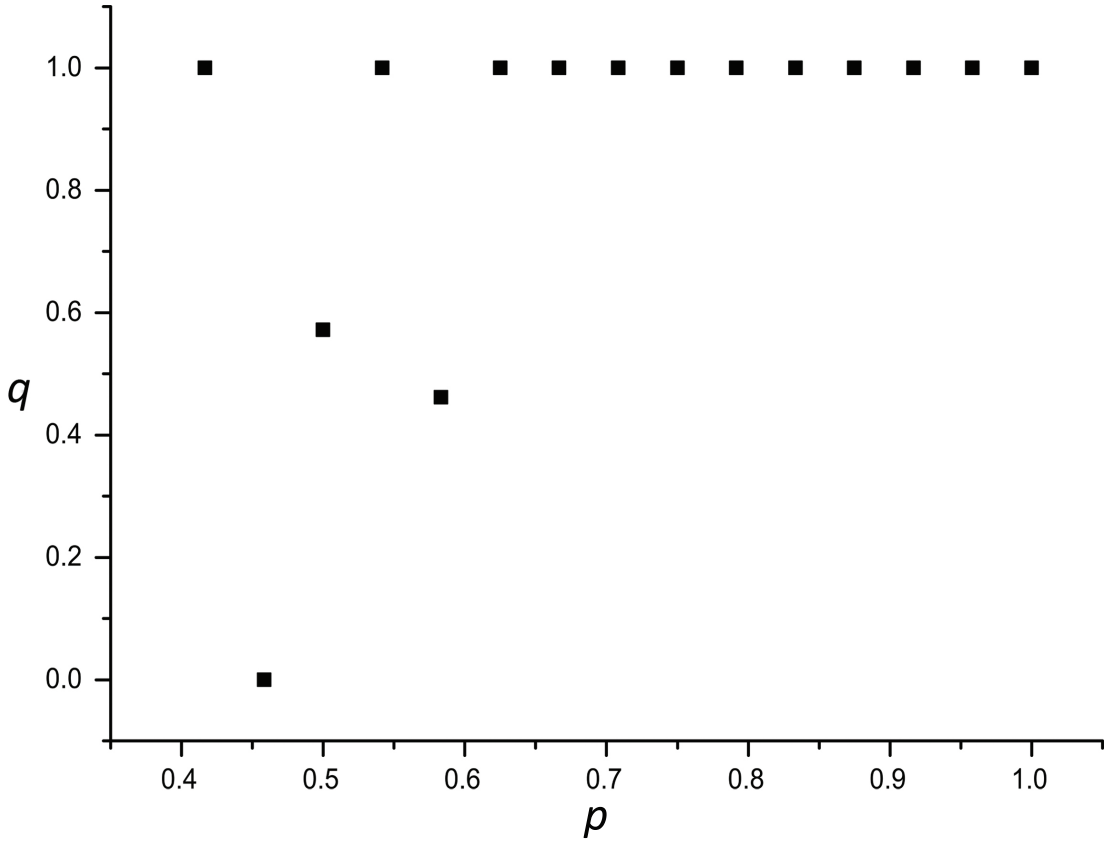


Figure 2.8: (Color online.) The vertical axis is the fraction (q) of two links from neighboring sites that impinge on a given node (j) found by the replicas that appear in the optimal (i.e., shortest tour) length solution. The horizontal axis is the probability (p_j) of finding this common set of two neighbors connected to the given node j; this probability p_j is identical to the vertical axis in Fig. 2.7. As this figure illustrates for sufficiently large values p_j, essentially all of the links found by many replicas also appear in the true optimal solution.

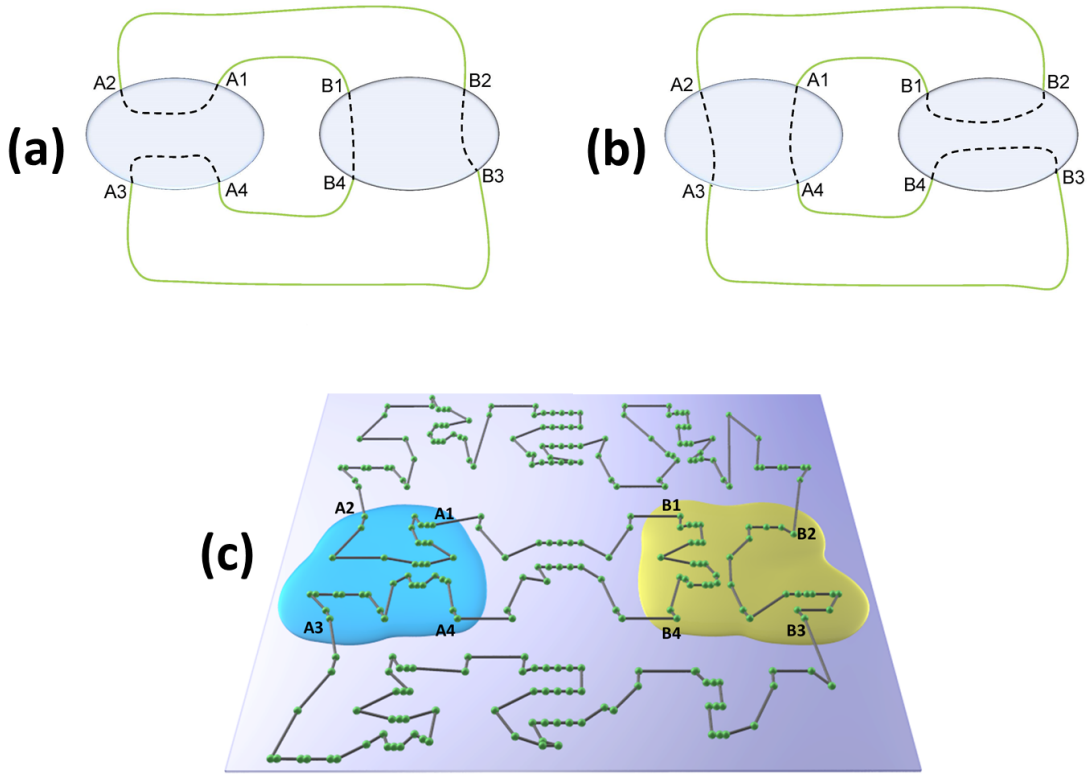


Figure 2.9: (Color online.) A schematic top view of Fig. 1.3. (a) one possible pairing inside the bubbles (b) the other possible pairing inside the bubbles. The green solid lines outside the blobs refer to the common structures shared by all of the 24 replicas while the dotted line inside the blobs denote the various possible bubble tours. The nodes (A1, A2, A3, A4, B1, B2, B3, and B4) are located on the boundaries of the shown bubbles. (c) a concrete example of (a).

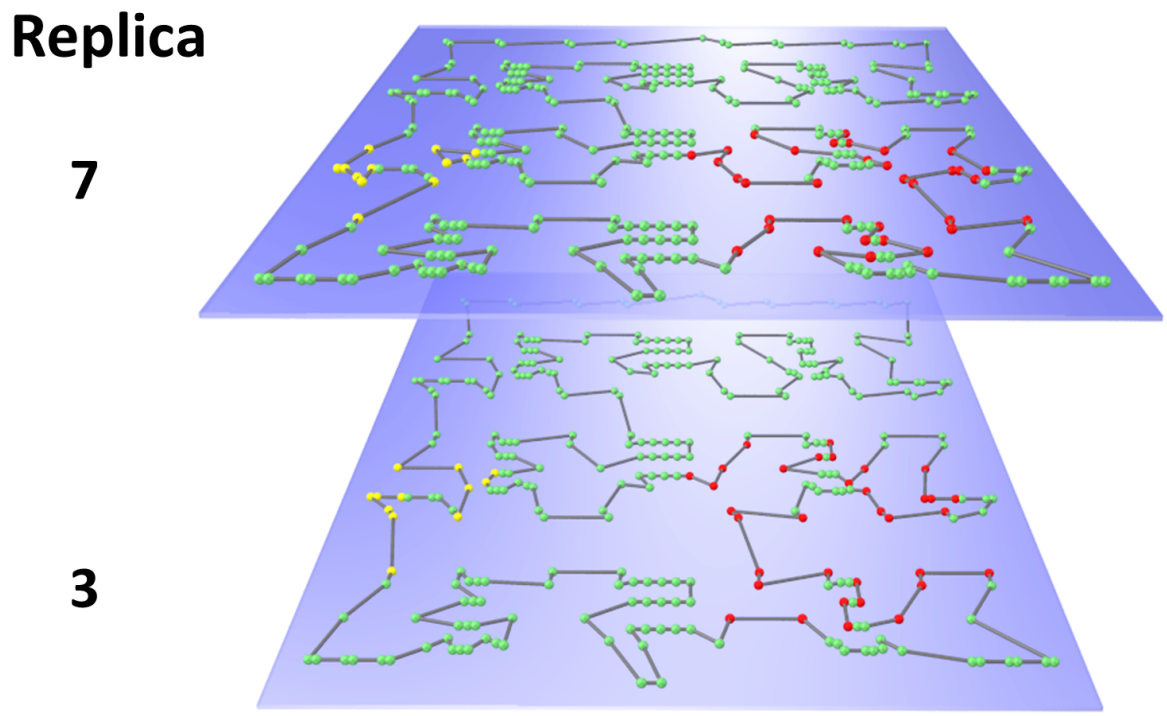


Figure 2.10: (Color online.) A specific illustration of how our method is applied to two particular replicas in our GDC algorithm in Sec. 2.3. The top place denotes the outcome of replica number 7 in our simulations while the bottom plane shows replica number 3. Green nodes are those nodes that have identical links in the set of all replicas (as in Fig. 2.6, 2.7). That is, nodes that are colored green have identical neighbors in all replicas. The remaining nodes with links that differ between the disparate replicas form separate "bubbles" attached to the backbone of common (green) nodes. We mark the nodes in the different "bubbles" by different colors. The yellow and red nodes form two bubbles where the tour configurations in the individual replicas differ. Amongst the two replicas shown, the shorter path in the bubble formed by the yellow nodes appears in replica 3. This intra-bubble configuration may be implemented in replica 7 to replace the original one shown. Once this transfer is done, the optimal tour (shown in Fig. 2.11) results.

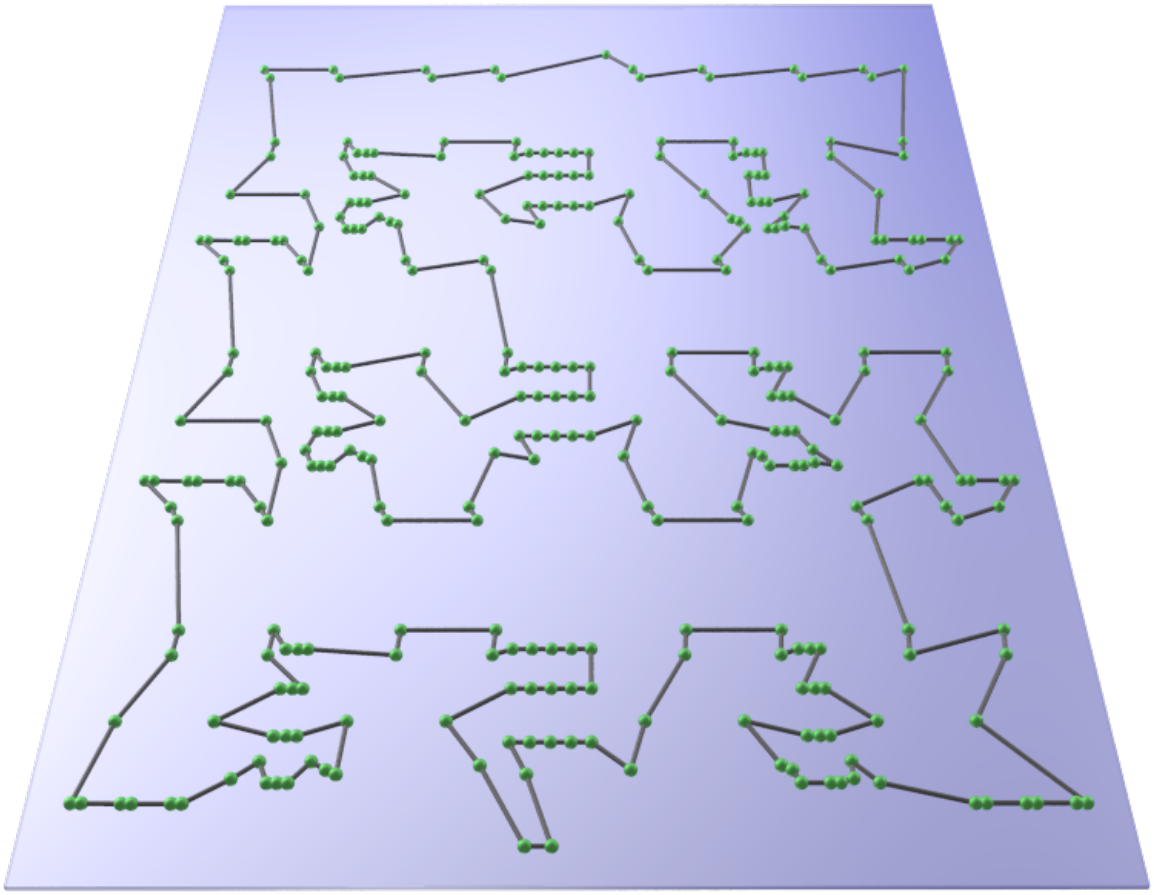


Figure 2.11: (Color online.) Optimal solution for lin318 from TSPLIB as discussed in Fig. 2.10. Following this transfer of the tour segment inside the bubble from replica 3 in Fig. 2.10, the new replica 7 attains the lowest distance optimal tour for the lin318 problem.

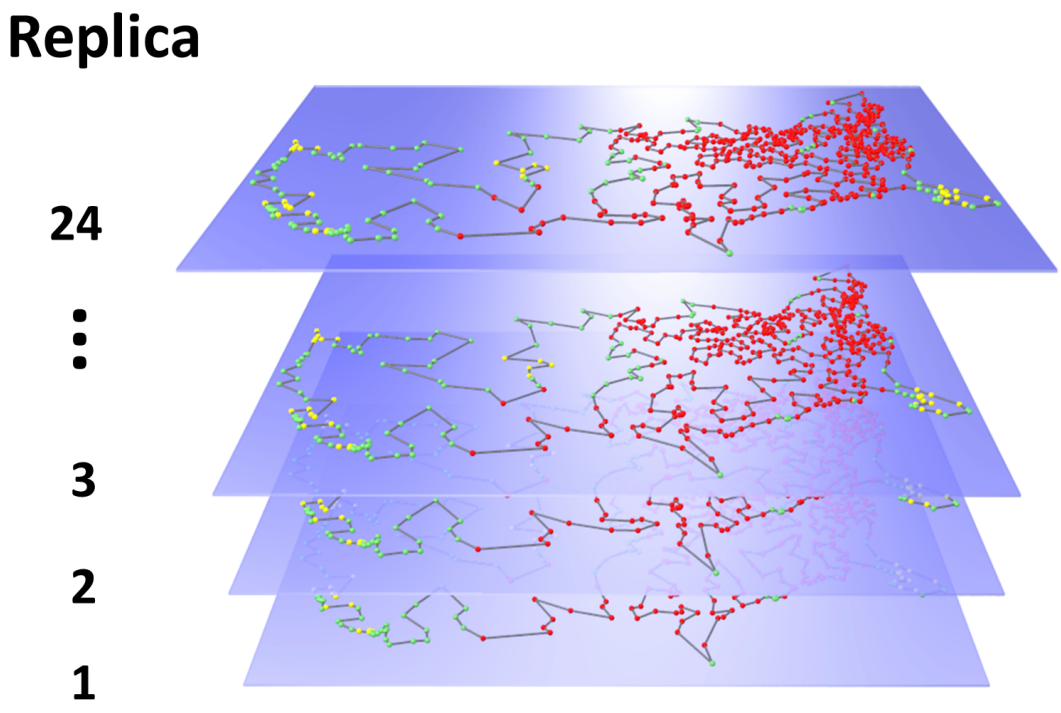


Figure 2.12: (Color online.) An all replica comparison that was used to produce the common (green) backbone of links for the 532 node att532 problem. In this example, we found a total of six bubbles attached to the common backbone. Five of these bubbles were of the "one-in-one-out" type (denoted yellow above). The nodes in the more challenging "three-in-three-out" type bubble are marked red.

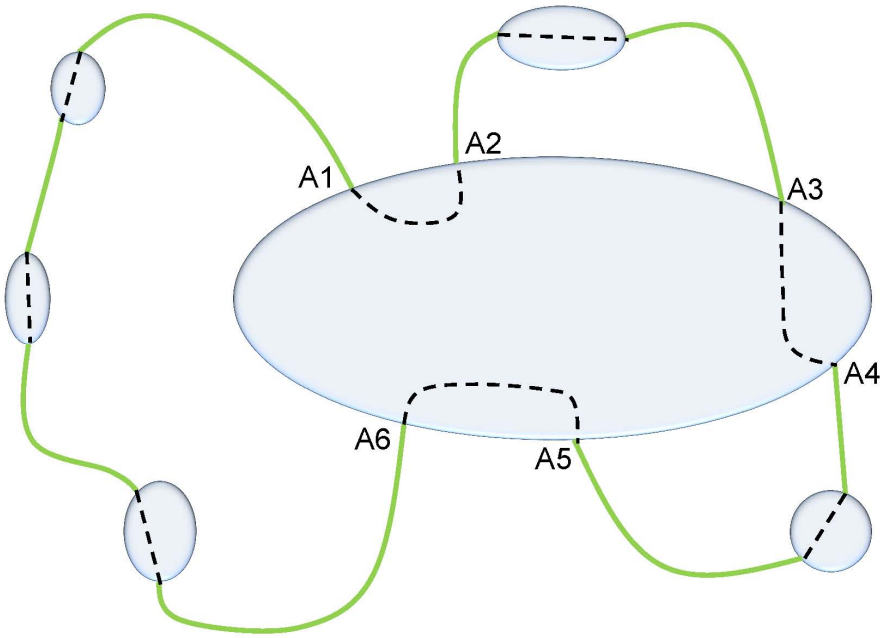


Figure 2.13: (Color online.) A schematic top view of Fig. 2.12. The green solid lines denote the common tour path between the replicas. The dotted line inside the blobs denote the possible various bubble tours. The nodes (A1, A2, A3, A4, A5, and A6) are situated on the periphery of the "three-in-three-out" bubble marked red in Fig. 2.12.

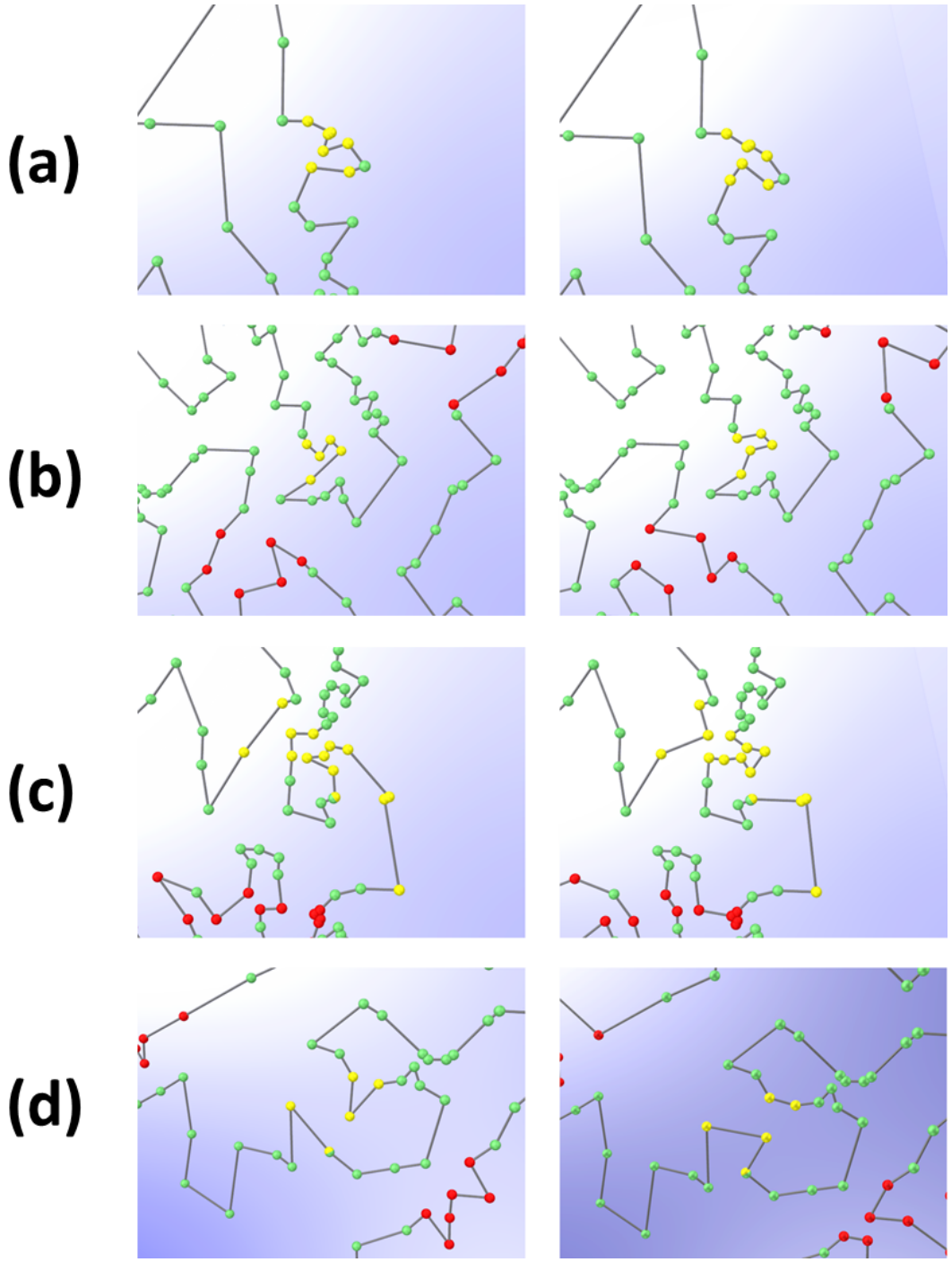


Figure 2.14: (Color online.) A comparison between replica 17 and other replicas in the att532 problem (see Figs. 2.12 and 2.13 for notation and color convention). (a) A side by side comparison between replica 17 (left) and replica 1 (right). Once the shorter intra-bubble path (marked yellow) from replica 1 is implemented in replica 17, the total tour length in replica 17 is reduced by a distance difference of size 8. (b) A similar comparison (for a region with another one-in-one-out "yellow bubble" different than that shown in panel (a)), between replica 17 (left) and replica 1 (right). Coincidentally, here also, swapping the intra-bubble tour in replica 17 with the shorter one found in replica 1 further lowers the total length by 8. (c) A further analogous comparison between replica 17 (left) with replica 10 (right). Replacing the initial other intra-bubble tour in replica 17 by the shorter one found for this bubble in replica 10 leads to a further lowering the tour length by 26. (d) A comparison between replica 17 (on left) with replica 23 (right) for a fourth "one-in-one-out" bubble in Figs. (2.12, 2.13). Using the shorter intra-bubble tour found in replica 23 instead of that initially found in replica 17 leads to a further reduction of the tour length in replica 17 by 14.

Bibliography

[1] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin glass theory and beyond*, (World scientific, Singapore, 1987), Vol. **9**.

[2] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin glass theory and beyond*, (World scientific, Singapore, 1987), Vol. **9**.

[3] A. Percus, et al., *Parallel tempering for the traveling salesman problem*, No. LA-UR-08-05100; LA-UR-08-5100. Los Alamos National Laboratory (LANL), (2008).

[4] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning." Addison Wesley (1989).

[5] P. Ronhovde and Z. Nussinov, Phys. Rev. E **80**, 016109 (2009).

[6] P. Ronhovde, S. Chakrabarty, D. Hu, M. Sahu, K. K. Sahu, K. F. Kelton, and N. A. Mauro, and Z. Nussinov, The European Physical Journal E **34** 1 (2011).

[7] P. Ronhovde, S. Chakrabarty, D. Hu, M. Sahu, K. K. Sahu, K. F. Kelton, N. A. Mauro, and Z. Nussinov, Scientific reports **2**, 329 (2012).

[8] D. Hu, P. Ronhovde, and Z. Nussinov, Phys. Rev. E **85**, 016101 (2012).

[9] D. Hu, P. Sarder, P. Ronhovde, S. Orthaus, S. Achilefu, and Z. Nussinov, Journal of microscopy **253**, 54 (2014).

[10] D. Hu, P. Ronhovde, and Z. Nussinov, Phil. Mag. **92** (4), 406-445 (2012)

[11] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. Onnela, Science **328**, 876 (2010).

[12] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin, Nature Physics **8**, 40-48 (2012).

[13] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. del Pozo, and S. Boccaletti, Scientific reports **3**, 1344 (2013).

[14] G. Bianconi, Phys. Rev. E **87**, 062806 (2013).

[15] R. C. Alamino, J. P. Neirotti, and D. Saad, Phys. Rev. E **88**, 013313 (2013).

[16] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, (Anchor Books, 2005).

[17] M. Clerc, *Particle Swarm Optimization*, (Wiley, 2006).

[18] M. Dorigo and L. M. Gambardella, BioSystems **43**, 73 (1997).

[19] M. Padberg and G. Rinaldi, SIAM Review **33**, 60 (1991).

[20] M. Grötschel and O. Holland, Mathematical Programming **51**, 141 (1991).

[21] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," in Local Search in Combinatorial Optimization (1997), pp. 215-310.

[22] S. Lin and B. W. Kernighan, Operations Research **21**, 498 (1973).

[23] K. Helsgaun, European J. Operational Research **126**, 106 (2000).

[24] M. Dorigo and L. M. Gambardella, Evolutionary Computation, IEEE Transactions **1**, 53 (1997).

[25] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

[26] Zakir H. Ahmed, "Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator", International Journal of Biometrics and Bioinformatics, **3**, 96 (2010).

Chapter 3

A Stochastic Replica-Based Voting Machine Algorithm For Supervised Learning

3.1 Introduction

Humankind is unequivocally living in the age of "Big Data". The rapid increase in connectivity between people, businesses and consumers, the media, and more has led to an explosion of publicly and privately available data. New information is constantly generated by social media, polling, market surveys, digital cameras, government surveillance, smart phones, scientific experimentation, and a multitude more of the technological sources and innovations of the past few decades. By the year 2020, the rate of production of digital data is projected to be 44 times as high as the rate in the year 2009, and the overall amount of available data is projected to be as high as 44 zettabytes (1 zettabyte $=10^{12}$ gigabytes). This wealth of information available in the digital ecosystem, combined with ever-increasing information storage capacity, has incredibly far reaching implications in diverse applications [2, 1, 5, 6, 4, 3]. In order

to realize the potential of the available data, methods for gaining meaningful insights must be developed. As the sheer quantity of available data exceeds human computational capability, efficient computer algorithms must be created and implemented. This is where the field of machine learning comes into play [7].

We can view machine learning as finding a function f which can describes the relationship between the input and output. The basic scheme consists of analyzing a set of input data ("training data") containing many entities (instances) to which we want to assign some value (label). Each instance is described by a set of quantities (features) which, theoretically, allow it to be mapped to a specific label. The problem, then, is to find a mapping algorithm (model) with parameters which the computer can fit to the given input data, and subsequently apply to future data ("testing data"). In this chapter we foucus on the supervised machine learning problem in which the labels are already known. Since the advent of supervised machine learning a number of algorithms have been developed. These are of varying complexity and performance, with some of the most popular being Support Vector Machine (SVM) methods [31, 32]. One may wonder why, in light of the plethora of currently available powerful methods, should we be concerned with the development of novel algorithms? Crudely, in addition to the benefits of having a robust "toolbox" of multiple algorithms, it turns out that existing algorithms are not without their faults.

3.2 Overview of Algorithm

During the last forty years, many of the developments in computer science have been spurred or influenced by topics and developments in the natural sciences. Indeed, artificial neural networks take as their basis, the biological networks of the brain, and have had tremendous success in advancing artificial learning [17, 33]. The study of spin-glasses by physicists and materials scientists over the last 15 years has led to the development of Hopfield networks and the addition of thermodynamic and statistical mechanics principles has to these networks has led to some of the most sophisticated

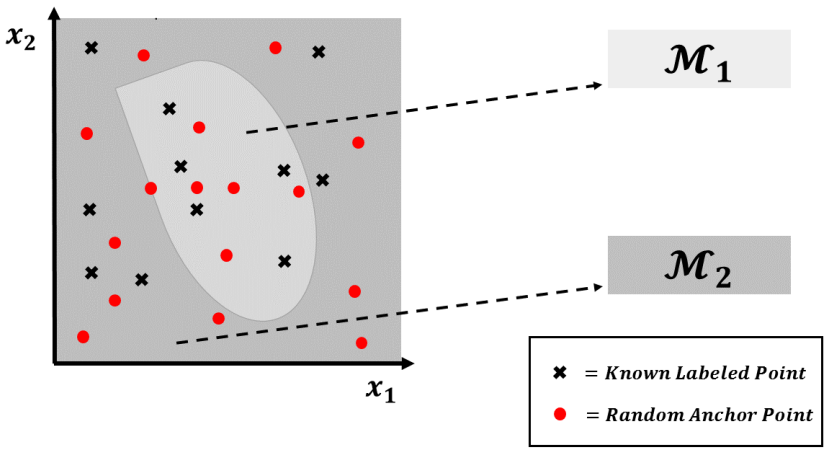


Figure 3.1: (Color Online.)'Phase space' representation of the feature space and the mapping of feature vectors to their respective classification label. Most variants of our algorithm rely on the use of 'Anchor points' (see Section 3.3).

learning models to date [19, 20, 21, 22, 23, 24]. It is evident that natural scientific principles (such as those from biology or physics) may serve as excellent bases for constructing learning algorithms. Additionally, recent results demonstrate that a certain theoretical basis may be required in order to enable learning algorithms to apply to scientific data [25]. With these ideas in mind, we have formulated a novel algorithm for supervised learning, which takes elementary statistical mechanics and phase transition theory as its motivation.

In classical statistical mechanics, e.g., [26] each atom carries its own phase space degrees of freedom: the atomic positions and momenta (thus in three-dimensional space, the state of each atom is defined by its three position and three momenta components- i.e., six degrees of freedom). At any given instant, the 'list' of all atomic coordinates and momenta for all the atoms in the system completely specifies its instantaneous state. Thus, for a system of N atoms, this 'microstate' can be represented as a single point in a 6N-dimensional phase space. The system itself, comprised of an extremely large number of particles, is macroscopic and can be described using only a few bulk degrees of freedom (i.e., temperature, pressure, magnetization, etc.). These bulk degrees of freedom characterize the observable state of the system in what is termed the 'macrostate'. The dynamical evolution of the atoms in

the system causes the microstate to constantly change, transitioning to new points in the phase space (new 'list' of 6N coordinates). If the system is in equilibrium, there is no change in macroscopic degrees of freedom with time, and this means that the microstates correspond in some way to the given macrostate. Additionally, the properties of the macrostate can be found by taking an *ensemble average* over the microstates corresponding to the macrostate. In general, changing external constraints changes the microstates that are available to the systems atoms, and the macrostate can also change. This implies that various sets of microstates correspond to specific macrostates, and this is indeed the case. More specifically, each microstate corresponds to only one specific macrostate. In the phase space picture, then, certain regions of phase space (corresponding to sets of microstates) will map directly to a single macrostate, and there will be boundaries in the phase space separating the different regions.

The above description of statistical mechanical phase space is analogous to what occurs in classification problems. As discussed in the introduction, classification-based learning problems consist of instances (the atoms) which are described by a set of features (positions and momenta). These values of the features for a given instance are cast into a feature vector which gives the 'location' of the instance in high-dimensional feature space (phase space). Each instance has an associated classification label corresponding to the set of features, such that certain regions of feature space map to specific labels. The goal of the learning algorithm is to find the boundary between the classification labels in feature space, so that new instances (which correspond to some point in feature space) can be appropriately mapped to the proper label. A schematic is provided in Fig. 3.1.

In order to achieve this goal, we need an appropriate mapping function $f(\vec{x})$, where $\vec{x}$ is a vector representing a particular point in the space of all d attributes ("features") go the data. In the statistical mechanical framework, mapping to a specific macrostate is done via minimization of an appropriate free energy. Once the free energy is properly extremized, calculating its value for a given point in phase space will allow for the elucidation of the corresponding macrostate or phase. Twentieth century physicist

Lev Landau studied free energies that could be expanded in a set of kernel functions of features (the so-called "order parameters"). The kernel expansion with coefficients whose values were fixed through optimization could then be applied to determine which macrostate a region of phase space belonged to [26, 27]. Thus, borrowing from this idea, since we are interested in identifying classification boundaries, we will assert that the label, y_i of a given instance can be expanded with unknown coefficients, in a set of kernel functions which take as their argument the feature vector. For a binary classification problem, the sign of a voting function weighted by different "replica" functions f determines the classification of the vector $\vec{x}$.

3.3 The SRVM algorithm in a nutshell

The model is trained using instances with known labels, but typically, the training data only covers a sparse set of the total feature space. To work around this, our algorithm uses a 'reverse ensemble averaging' technique that randomly samples the feature space. We generate a stochastic set of v feature vectors and associated feature space points, which we call 'anchor points' and then use their proximity to training points to assign a classification label. Essentially, we use the known labels corresponding to training points in feature space (instances) with sufficiently localized kernel functions to attempt to classify the space around the known points so as to create general mapping functions. We consider a specific input "training" data of size N points each comprised of d features (expressed as a $d-$ dimensional vector) for these points $\{\vec{x}_i\}_{i=1}^{N}$ and the corresponding given correct classification $\{y_{i,c}\}_{i=1}^{N}$. With these, we will define

$$y_{i,p}^{\alpha} \equiv f^{\alpha}(\vec{x}_i) \equiv \sum_{j=1}^{v} c_j^{\alpha} \; K(\vec{x}_i, \{\vec{\chi}_j^{\alpha}\}), \tag{3.1}$$

and aim, as we will describe below, to set $y_{i,p}^{\alpha}$ equal to the known correct classification $y_{i,c}$. Here, $\{\vec{\chi}_j^{\alpha}\}_{j=1}^{v}$ are fixed random vectors (which we will often term "anchor

vectors") that are different for each "replica" α, and K is a stochastically chosen function. It may, e.g., be any of the below standard functions,

$$K(\vec{x}_i, \vec{\chi}_j^{\alpha}) = \begin{cases} \exp\left(-\frac{(\vec{x}_i - \vec{\chi}_j^{\alpha})^2}{2\sigma^2}\right) \\ \exp\left(-\frac{|\vec{x}_i - \vec{\chi}_j^{\alpha}|}{\xi}\right) \\ \left(\frac{|\vec{x}_i - \vec{\chi}_j^{\alpha}|}{\gamma}\right) \\ \left(a|\vec{x}_i - \vec{\chi}_j^{\alpha}|\right) \\ \frac{1}{1+\exp\left(q|\vec{x}_i - \vec{\chi}_j^{\alpha}|\right)} \end{cases}, \tag{3.2}$$

where σ, ξ, γ, A, and q are constants that serve as defining parameters for the (Gaussian, exponential, complementary error function, Airy functions (of the first kind), and Fermi type distribution) functions that appear above. As we will further explain, in Eq. (3.1), $\{f^{\alpha}\}_{\alpha=1}^{\mathcal{R}}$ is a set of viable functions of the variables $\vec{x}$ (different specific functions (either of various types (Eq. (3.2)) or, more commonly in our simplest analysis, functions of a certain general type having yet different fixed vectors $\{\vec{\chi}_j^{\alpha}\}$) associated with "different replicas" α) . We may trivially re-express the above as $\vec{y}^{\,\alpha} = \underline{K}^{\alpha}\vec{c}^{\,\alpha}$ where $K_{ij}^{\alpha} = K(\vec{x}_i, \vec{\chi}_j^{\,\alpha})$. In statistical mechanics, the coefficients in the expansion function are found by minimizing the free energy, $\mathcal{F}$ (possibly subject to additional constraints as will be discussed later). We may invert the latter linear equation,

$$\vec{c}^{\,\alpha} = (\underline{K}^{\alpha})^{-1}\vec{y}_c, \tag{3.3}$$

where $\vec{y}_c$ is the vector (with components $y_{i,c}$) of correct classification results and $(\underline{K}^{\alpha})^{-1}$ is the inverse of the Kernel matrix. With the aid of Eq. (3.3), we solve for the coefficients c_n^{α}. Typically, the systems that we study are underdetermined. Therefore, the inverse matrix $(\underline{K}^{\alpha})^{-1}$ is actually a pseudo-inverse; finding the coefficients c_n^{α}

involves a least squares fit. For each replica α, we minimize a "learning free energy"

$$\mathcal{F}^{\alpha} = \sum_{i=1}^{N} (y_{i,p}^{\alpha} - y_{i,c})^2. \tag{3.4}$$

Here, $y_{i,p}^{\alpha}$ are the predicted (p) results (as given by Eq. (3.1)) while $y_{i,c}$, as noted above, are the replica independent correct (c) classification results that a good algorithm should aspire to uncover. Thus, the coefficients c_j^{α} that are calculated for a given replica α will appropriately map a given "state" $\vec{x}$ to the correct "phase" label given the phase space sampling information. We repeat the above calculation for multiple stochastic sets of replicas ($\mathcal{R}$ in total) in an attempt to "reverse ensemble average" based on knowledge of the actual phase space mapping to appropriately find the correct divisions. As the mapping functions f are continuous while the classification labels are discrete, the output of the mapping function for each replica has to be thresholded. Once the system is "trained" (i.e., the coefficients c_j^{α} are fixed by Eq. (3.3), we examine what occurs for new "test" input vectors $\vec{x}$. For the binary classification cases that will be studied throughout this chapter, we will typically set, for each replica α,

$$y^{\alpha}(\vec{x}) = \begin{cases} -1 & f^{\alpha}(\vec{x}) < 0 \\ 1 & f^{\alpha}(\vec{x}) \geq 0 \end{cases}. \tag{3.5}$$

This thresholding can be generalized for multi-class classification by varying the threshold point, and in general Receiver Operating Characteristic (ROC) curves [28] can be used to test for the best value of the threshold. Once the output is calculated for all points in each of the replicas, the **overall classification** of an instance is found via **voting**. The "overall" label of a given instance is found by taking the average of the values predicted for that instance across all replicas, and then appropriately thresholding it (as in Eq. (3.5)). That is,

$$\mathcal{V}(\vec{x}) = \frac{1}{\mathcal{R}} \sum_{\alpha=1}^{\mathcal{R}} y^{\alpha}(\vec{x}), \tag{3.6}$$

where $y^{\alpha}(\vec{x})$ is the predicted label by the α-th replica.The process of voting based on stochastic replicas allows for the correction of occasional mislabeling due to random fluctuations, and leads to a more reliable final result.

The specific, equal weight, voting of Eq. (3.6) is one of many possible voting choices that may be employed. As we will discuss later on, that multiple voting methods could be used to increase the overall performance. Since the averaging implicit in voting leads to a continuous range of voting outcomes, the same thresholding methodology of Eq. (3.5) will be employed. We contrasted our results with those found by SVM.

3.4 Evaluation of the SRVM Algorithm

To assess the performance of the SRVM algorithm, we will apply it to several test datasets and examine various statistical performance metrics.

To use the SRVM algorithm to fit a model to the data in the datasets considered, we broke the data up into a training set and a testing set. The training set was used to fit the model (i.e., solve Eqs. (3.1,3.4)), and the testing set was used to evaluate the performance of the model. Some of the datasets employed in testing the SRVM algorithm that are discussed in this chapter came with explicit testing datasets. For other data sets no explicit test set was provided; in these cases, we applied five-fold cross validation (CV) techniques to fit and analyze the model. Five-fold CV involves randomly splitting the dataset into five equal size subsets or folds, and using 4 of these folds together as a training set and the fifth fold as a testing set, while iteratively cycling through so that each fold serves as the testing set once. This allowed us to analyze the performance of the model for multiple folds, as well as report average performance metric values across all five folds. This five-fold CV was used throughout to ascertain the accuracy. Unless explicitly noted otherwise, all accuracies that we report were obtained by five-fold CV.

Throughout, we used various kernels K (Eq. (3.2)) when performing the expan-

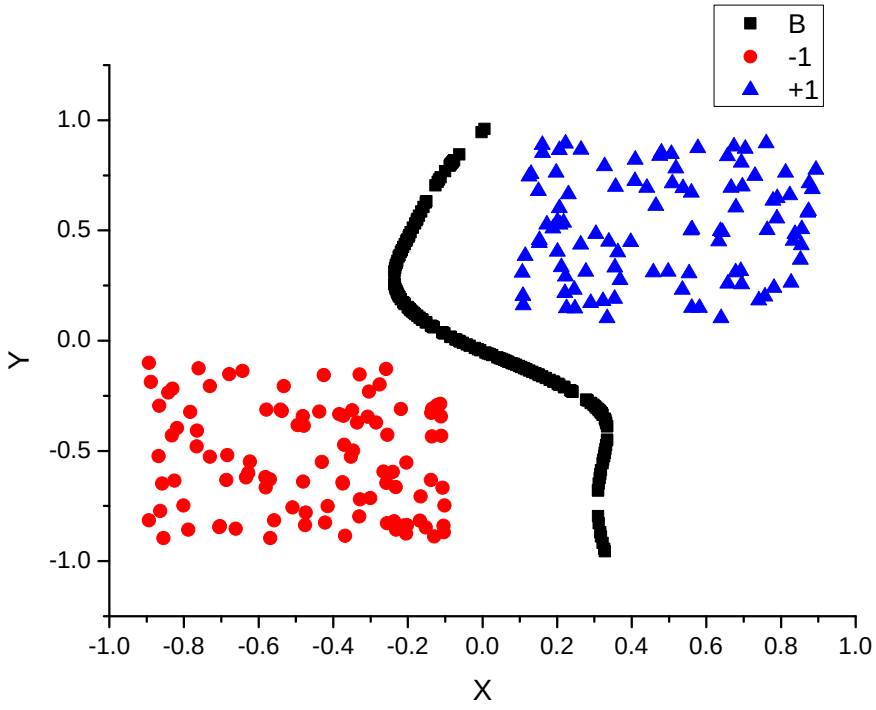


Figure 3.2: (Color online.) The boundary formed by Gaussian Kernel algorithm in the linearly separable case.

sions of Eq. (3.1). We further examined multinomial forms of particular maximum degrees n_k in each single feature x_k,

$$f^{\alpha}(\vec{x}) = \sum_{m_1=0}^{n_1} \sum_{m_2=0}^{n_2} \cdots \sum_{m_d=0}^{n_p} c_{m_1 m_2 \cdots m_d} \prod_{k=1}^{d} x_k^{m_k}. \tag{3.7}$$

Here, $\{c_{n_1 n_2 \cdots n_d}\}$ are constants that may, similar to Eq. (3.1), be determined by Eq. (3.3) (the minimization of Eq. (3.4)).

We next explicitly turn to the examples that we tested.

• Our first test case is that of our own synthetic data that allow for a simple linear separation between two sets with non-intersecting convex hulls (the two sets appear in the upper right and lower left sides in Fig. 3.2). The goal of the algorithm is to detect this structure and correctly classify different points as belonging to either of these two data sets. We used Eqs. (3.1, 3.3) with a Gaussian kernel K for $v = 50$ fixed vectors $\{\vec{\chi}_j^{\alpha}\}$ that were randomly chosen for each of the $\mathcal{R}$ replicas; this led

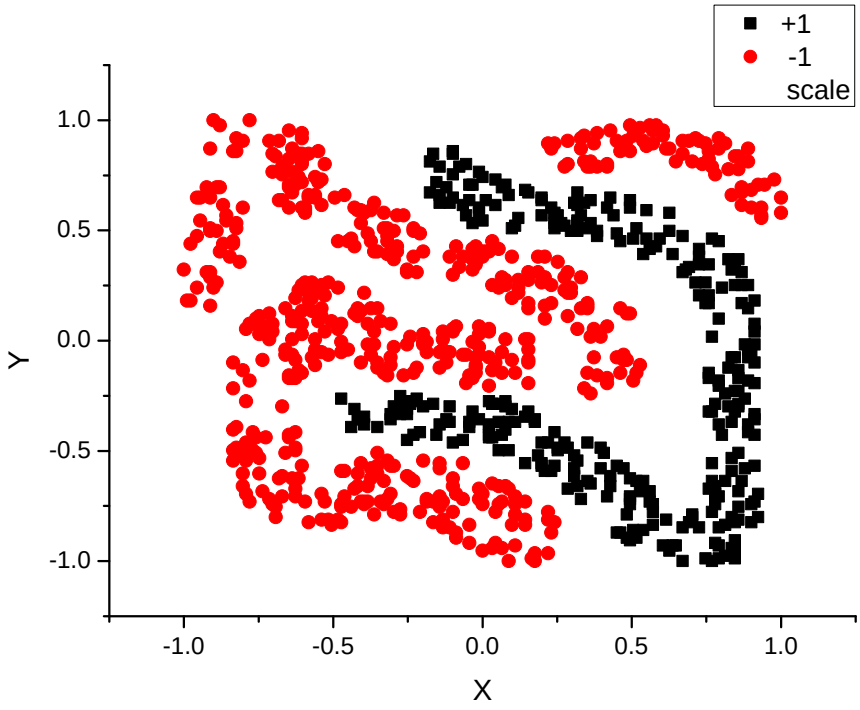


Figure 3.3: (Color online.) The raw data of the Four-class problem.

to an accuracy (as ascertained by the 5-fold CV) of 100%. Fig. 3.2 illustrates the distribution of the two data sets and the boundary formed by the Gaussian Kernel SRVM algorithm. The boundary obtained by our method is a smooth surface- not a straight line as found by other class classification algorithms that we tested (e.g., SVM with a linear kernel, logistic regression, and other linear classifiers); the linear kernel SVM algorithm similarly achieved an accuracy of 100%.

In the remainder of this chapter, we will focus on far more pertinent non-linearly separable benchmarks.

• The next data that we test is that of the "Four-class" [30] benchmark- a binary classification problem having $d = 2$ features for each of its 862 data points. Fig. 3.3 visually depicts the data on a $d = 2$ dimensional map. Similar to our first example, the goal of the machine learning algorithm is to correctly identify the binary classification of input data (similarly set to be +1 (marked black in Fig. 3.3) or -1 (red)). We obtained a perfect (i.e., 100%) accuracy when applying SVM with a radial kernel. We studed this system with our SRVM method with the multinomial kernel of Eq. (3.7).

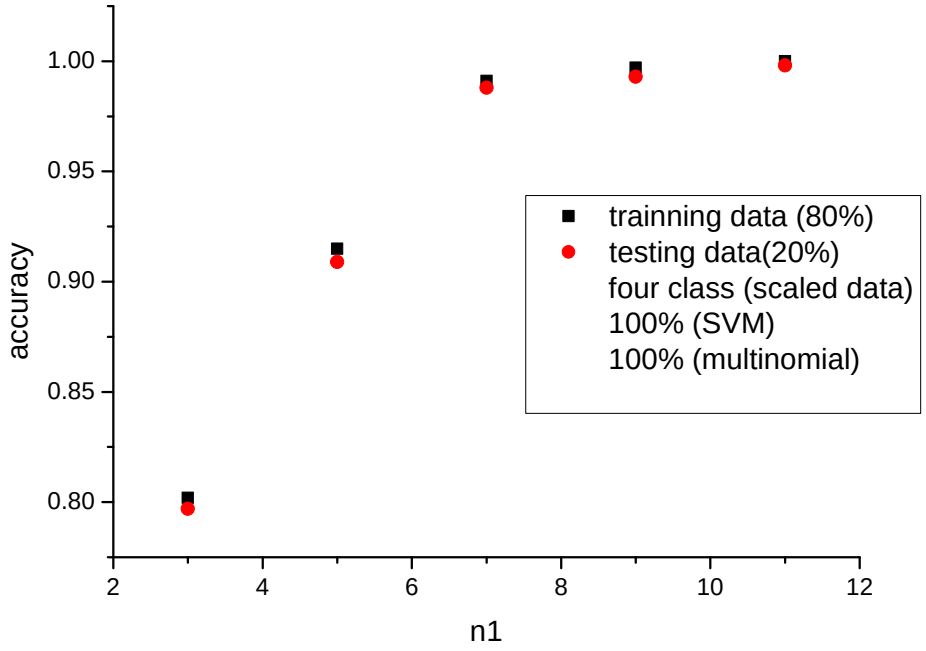


Figure 3.4: (Color online.) The accuracy of the multinomial variant of our SRVM algorithm for the Four-class problem. Here, n_1 denotes the highest power (of any of the features x_k) in the multinomial expansion of Eq. (3.7).

Fig. 3.4 demonstrates how the prediction accuracy varies with the multinomial order $n_1 (= n_2 = \cdots = n_d)$. In the tested range, is monotonic with increasing polynomial order. When the multinomial order $n_1 = 11$, the accuracy is 100 %. Figures 3.5, 3.6, and 2.9 provide the boundaries found when n_1 equals 3, 5 and 7 respectively. We see that when $n_1 = 7$, a smooth boundary between the two classes results. We similarly applied our algorithm with a Gaussian kernel K to the Four-class problem. We first discuss the single replica results. The number of fixed vectors v in Equation 3.1 plays a important role in predicting the results. We initially randomly produced $v = 50$ fixed vectors (less than a tenth of the number of data set points). This led to an average accuracy of 99.09%. Reducing the number of fixed vectors to only $v = 10$ resulted in an accuracy decrease to 81.18%. In this and other instances, we saw that (not unexpectedly) when the number of fixed vector became too small, the prediction accuracy diminished. In Section 3.4.1, we will discuss this trend in greater

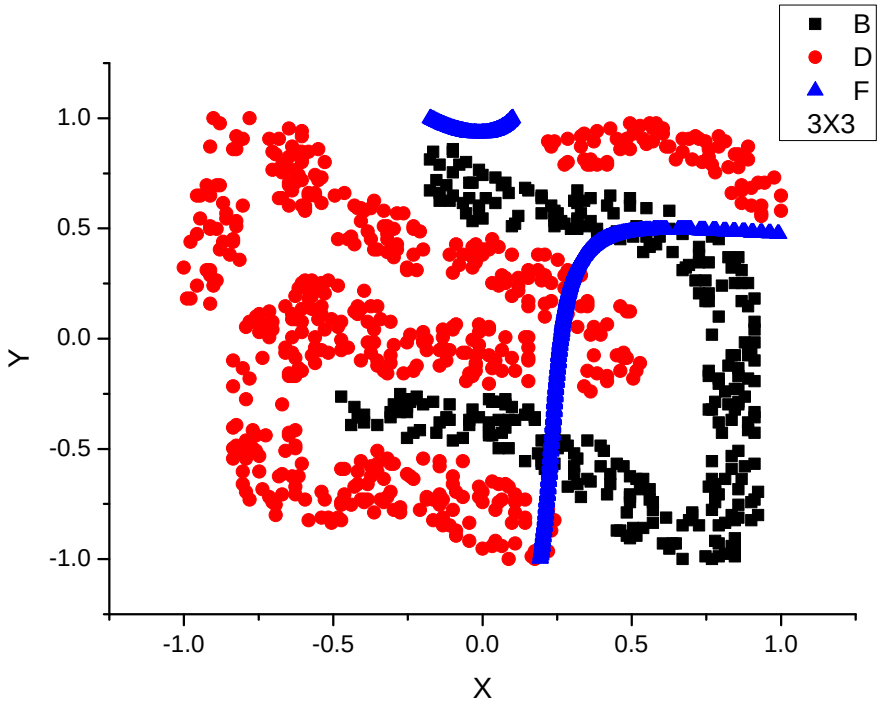


Figure 3.5: (Color online.) Boundary predicted by the multinomial kernel algorithm for the Four-class problem when $n_1(= n_2) = 3$. The signifiers B and D represent the +1 and -1 classes respectively. F denotes the boundary between these two classes as determined by the SRVM to this cubic order.

depth. As discussed in Section 3.3, the SRVM combines the single replica results via voting (Eq. (3.6)). To avoid a gridlock when performing such a vote, we chose the number of replicas $\mathcal{R}$ to be an odd number (we picked $\mathcal{R} = 7$ here). Each replica α corresponds to a possible predictor y^α that is related to a different set of fixed vectors $\{\vec{\chi}_j^\alpha\}$. Averaging over replicas (Eq. (3.6)) produced an accuracy of 99.76%.

• The subsequent test case is that of "Svmguide1" benchmark [30]. This well studied benchmark problem (originating from astroparticle physics) consists of training file and testing file (i.e., there is no need to perform CV). The number of data points in training file and testing file are, respectively, 3089 and 4000; each data point has $d = 4$ features. Optimizing and using the best parameters for a radial basis SVM kernel enabled a 96.9% accuracy. We applied our SRVM algorithm with a polynomial kernel (see Fig. 3.8) to this benchmark. Contrary to the Four-class problem, the accuracy initially grew with increasing polynomial order n_1; however, at larger n_1

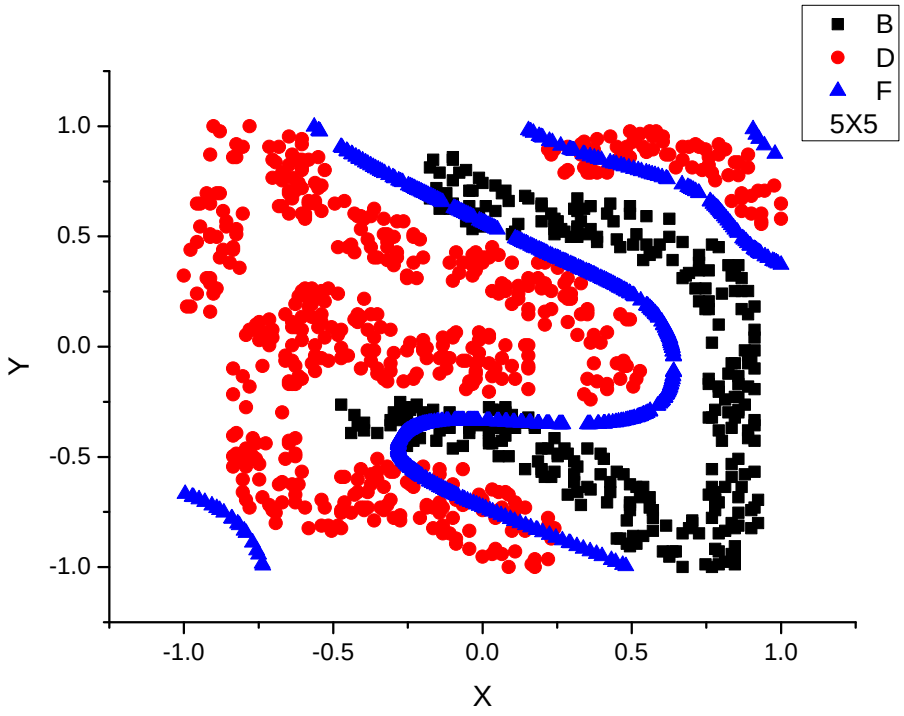


Figure 3.6: (Color online.) Boundary predicted by polynomial kernel algorithm for the Four-class problem when $n_1 = n_2 = 5$ in Eq. (3.7). As before, B and D mark the +1 and -1 classes respectively. The curve F denotes the boundary found by the SRVM algorithm between these two classes to this quintic order.

the accuracy diminished. The peak prediction accuracy for the test data is 96.6%. In Section 3.4.4, we will discuss how the best value of n_1 may be ascertained from replica overlap (without being given the results for the test data). We further applied the Gaussian kernel algorithm to the Svmguide1 problem and tested three different value of number of fixed vectors ($v = 50, 100, 200$). In single replica tests, the highest accuracy (95.6%) was realized for $v = 100$ fixed vectors. Setting $v = 50, 200$ gave rise to accuracies of 94.62% and 94.98% respectively. Using $\mathcal{R} = 7$ replicas in the Gaussian kernel algorithm, mproved the accuracy to 95.8%.

• The "Liver disorder" data set [30] is a benchmark problem that has 345 data points which has $d = 6$ features for each input. It has no testing file so that we performed the CV tests as before. We first investigated the performance of SVM. Optimizing the SVM parameters in a radial basis enabled an average CV accuracy of 71.88%. Next, we applied the ($n_1 = \cdots = n_6 = 3$) multinomial SRVM. This led

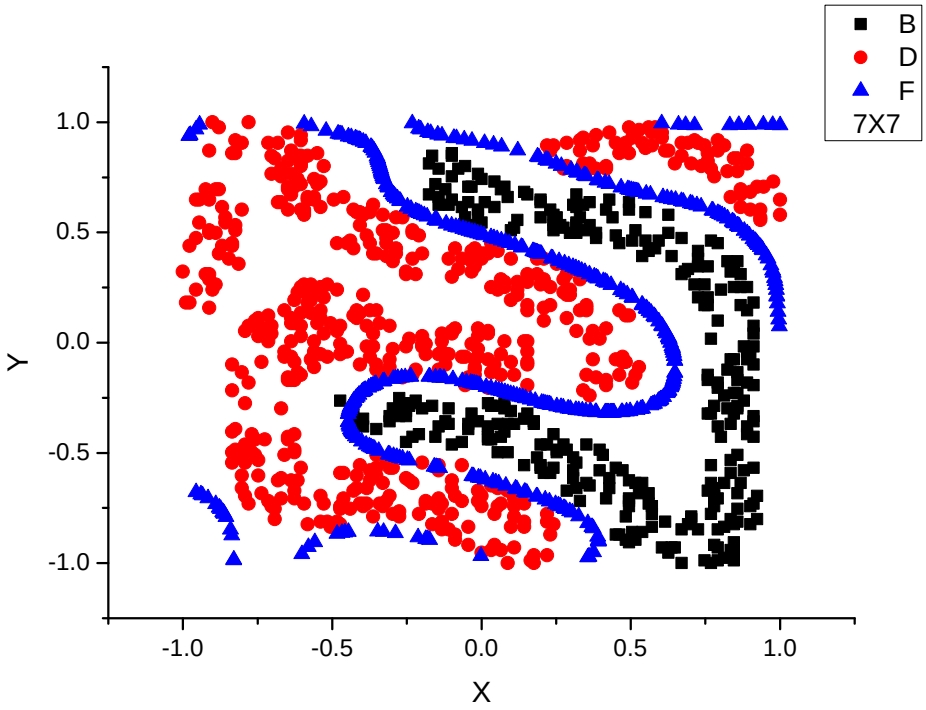


Figure 3.7: (Color online.) B and D represents the +1 and -1 classes respectively in the Four-class data set. The curve F denotes the boundary between these two classes to this Boundary predicted by the SRVM algorithm when using a multinomial of order $n_1 = n_2 = 7$.

to an average CV accuracy of 65.5%. Lastly, we applied the Gaussian kernel SRVM algorithm to the problem. We found the optimal number of fixed vectors is $v = 80$. This led, for the single replica variant, to an accuracy of 66.29%. We then couple different replicas ($\mathcal{R} = 7, 15$, and 21). The results illustrate that replica voting indeed improves the accuracy. Specifically, $\mathcal{R} = 7$ replicas led to an accuracy of 68.40%. In the case of $\mathcal{R} = 15$ replicas, we achieved an accuracy of 66.97%. For $\mathcal{R} = 21$, the average CV accuracy became 68.40%.

• As another example, we also tested the Heart disease from UCI machine learning repository database [31]. This is a binary classification problem consisting of 270 data points with $d = 13$ features. For calculations in this chapter, the data will be scaled (to be between $[-1, 1]$). We will present various aspects of our results for this prominent benchmark in later sections.

• The results from the Statlog Australian Credit Approval dataset [32] (hereby

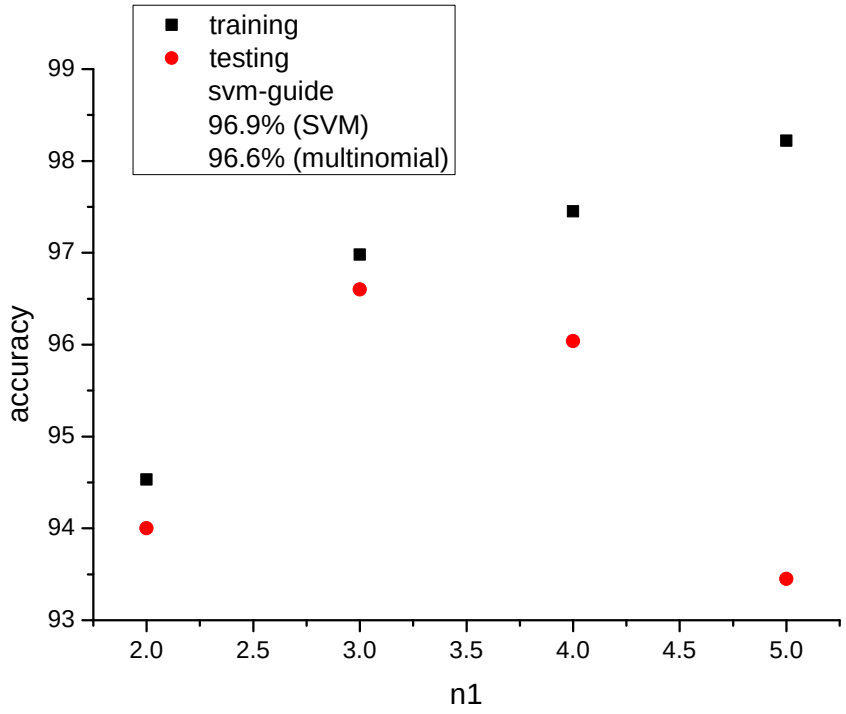


Figure 3.8: (Color online.) The accuracy and training set accuracy (the ability of the kernel to reproduce the training data) when using different order multinomial kernels (Eq. (3.7)) in the SRVM algorithm when applied the Svmguide1 data set. For comparison, we provide in the top panel, the optimal result found the SVM algorithm.

abbreviated to "Australian") will, similarly, also be presented. This benchmark is comprised of 126 binary-classified instances with 309 features and, as we will demonstrate, possesses characteristics which make it an excellent representative dataset.

When analyzing datasets using classification or regression algorithms, it is important to begin by preprocessing the data to be studied. In many datasets, it is common to have various instances which are missing values corresponding to certain features. Numerous methods exist to deal with missing values through various types of imputation [33, 34]. Typically the act of imputing data for missing values is itself a learning step, which inherently adds complexity to the analysis process. In the datasets studied here, the number of instances with missing values was small enough that these instances were discarded.

In addition to handling missing values, the preprocessing step also typically in-

volves scaling of the data, so that the values corresponding to a given feature are of the same scale as all of the other features. This suppresses any effects of a feature with high variance and magnitude, dwarfing features with smaller variances and magnitudes. The three main feature scaling types are normalization of the range of values for a given feature such that they have a mean of zero and variance of one, scaling to the range [0,1], and scaling to the range [-1,1]. In Section 3.4.3, we will test whether there is any statistically significant difference in the performance of the algorithm with different feature scaling types, but will employ normalization scaling unless otherwise noted. The data presented for the Australian data set are scaled to [-1,1].

3.4.1 Accuracy dependence on replicas and anchor vectors

When evaluating the performance of a binary classification model, the first step is typically to measure the accuracy of the classifier when applied to the testing data of known labels. The accuracy is simply defined as the percentage of correctly labeled instances in the testing set. In analyzing the LSVT voice rehabilitation dataset [29], we primarily used the Gaussian kernel of Eq. (3.2). A priori, the spread (σ) of this Gaussian may assume any value. We observed that setting $\sigma = \sqrt{N_{features}}$ yielded the best results. Consequently, this was the value used in our analysis. We employed the five-fold CV and examined the average accuracy, $\bar{A}$, across all five folds for various numbers of anchor points (v) and replicas ($\mathcal{R}$). The results of this analysis are presented in Fig. 3.9. In panel (3.9a), we show a 3D surface plot of the average accuracy as a function of the number of anchor points and number of replicas. In panels (3.9b) and (3.9c), we show projections of the 3D plot for constant v and $\mathcal{R}$, respectively. It is evident from these plots that the accuracy quickly reaches an asymptotic value with increasing replica number. Once a maximum is reached, further changes in the number of replicas have little net impact on the accuracy. Additionally, it is evident that (regardless of the number of replicas $\mathcal{R}$ used) the accuracy increases rapidly with number v of anchor points, levels off at a maximum, and then decays with further increasing v. The decay of average accuracy with increasing v beyond a

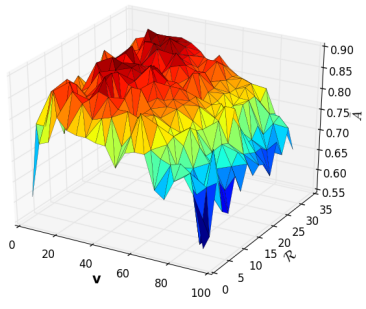
certain value is indicative of over-fitting. Analysis of the accuracy data presented in Fig. 3.9 suggests that a maximum accuracy of $\bar{A}$=88.9% for the LSVT dataset occurs at $v = 35$ and $\mathcal{R} = 29$.

We now turn to a similar analysis for the "Heart" dataset [31]. In order to find the optimal number of anchor points for these data, we changed the number of anchor points v from 10 to 250 in increments of 10 (see panel (a) of Figure 3.10). The resulting accuracy was averaged over 10 different sets of $\mathcal{R} = 31$ replicas analyzed with a 5 fold CV. The highest accuracy was achieved when $v = 40$. A further minimum in the accuracy appears for $v \sim 220$ anchor points. For anchor points as low in number as $v = 12$, our procedure yields an accuracy above 80% (a value quite close to the highest obtained accuracy of 82% that we obtained when using $v = 50$ anchor points). In panel (b) of Figure 3.10, we show the effect of increasing replica number on the average accuracy in Heart example. The range of the number of replicas is quite wide, $11 \leq \mathcal{R} \leq 201$. Both curves in this panel (corresponding to the average accuracy and the replica overlap) display an oscillatory behavior about the averaged result and the amplitude of oscillations decreases as the number of replicas $\mathcal{R}$ increases. Already for $\mathcal{R} = 31$ replicas, we achieved an average accuracy of 82%. Considering that the highest accuracy the we reached (as is seen in the graph) is 83.3% for $\mathcal{R} = 51$ replicas, in further analysis of the Heart dataset, we used the more modest number of $\mathcal{R} = 31$ replicas.

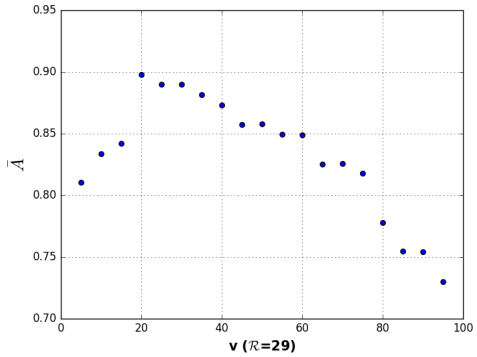
An important point that we will underscore and reiterate throughout this work (and discuss, more specifically, in Section 3.4.4), is that we may determine the optimal number of replicas $\mathcal{R}$, number of anchor points v, and any other undetermined quantity by noting when the average inter-replica is (near) maximal as a function of these parameters.

We return to our analysis of the Australian dataset. The dependence of accuracy on number of anchor points is tested on the Australian dataset with Gaussian kernel models with replica number $\mathcal{R} = 5$. Each point of the plot is the average of 20 randomly generated models; see Fig. 3.11(a).

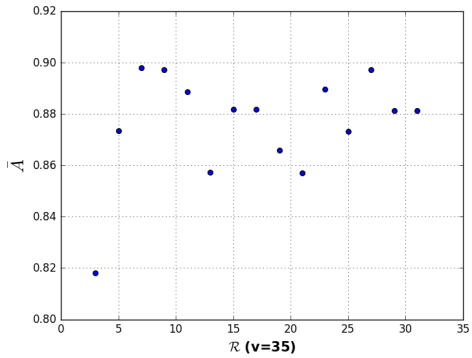
We observe that as the number v of fixed vectors is increased, initially the fitted



(a)



(b)



(c)

Figure 3.9: (Color Online.) Tests of the accuracy of the SRVM algorithm (with a Gaussian kernel) to the LSVT data set. (a) 3D surface plot of the average accuracy (ascertained by 5 fold CV) as a function of the number of anchor points, v, and the number of replicas, $\mathcal{R}$. (b) The 2D projection of this 3D plot (a) into the accuracy-anchor point plane with constant replica number, $\mathcal{R} = 29$ to show accuracy as a function of number of anchor points. The accuracy initially increases with more anchor points; beyond a threshold maximum value at $v = 20$, the accuracy drops (due to overfitting). (c) A projection of accuracy surface of (a) into the accuracy-replica plane with constant number of anchor points, $v = 35$ in order to highlight the dependence of the accuracy on the number of replicas. The accuracy initially rises, very rapidly, with an increase of the number of replicas and then nearly saturates.

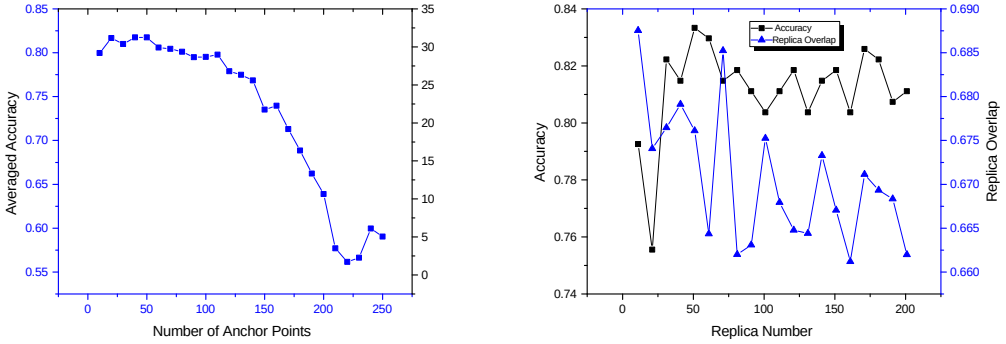


Figure 3.10: (Color Online.) Accuracy of the SRVM algorithm (with a Gaussian kernel) when applied to the Heart dataset. (a) Graph of the average accuracy as a function of the number of anchor points v and number of replicas $\mathcal{R} = 31$. (b) Plot of the average accuracy as a function of the number of replicas, $\mathcal{R}$ when the number of anchor points is held fixed at $v = 50$. In Section 3.4.4 we will define and analyze inter-replica overlaps (the one plotted here is the normalized variant of Eq. (3.10)). As seen here, the average replica overlaps correlate with the accuracy of the predictions.

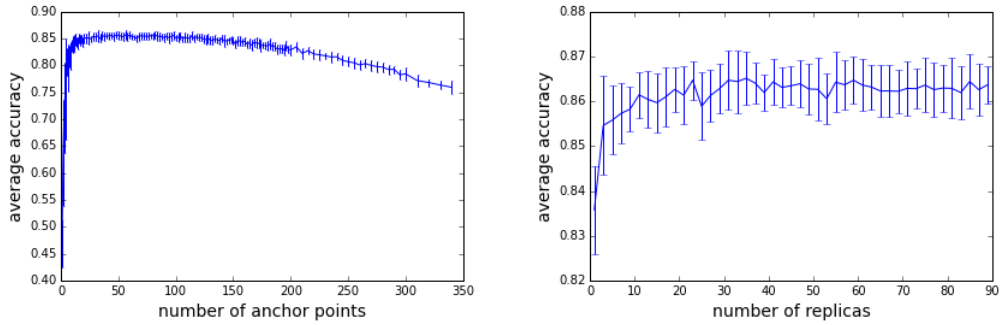


Figure 3.11: (Color Online.) Accuracy tests for the Australian dataset. We show a (a) Plot of average accuracy as a function of the number of anchor points v for a fixed small number of replicas ($\mathcal{R} = 5$), and (b) plot the average accuracy as a function of the number of replicas, $\mathcal{R}$ for $v = 50$ anchor points.

Data Set	Classes	Number of Instances	Number of Features	SVM	SRVM
LSVT	2	126	309	0.873	0.889
Advertisement	2	3279	1558	0.973	0.963
Australian	2	690	14	0.853	0.863
Heart	2	270	13	0.825	0.824
Four-class	2	862	2	1.00	1.00
Svmguide1	2	3089	4	0.969	0.966
Liver-disorders	2	345	6	0.718	0.684
Breast-cancer	2	683	10	0.942	0.945

Table 3.1: Summary of the Optimized Accuracy for both (a) the standard SVM algorithm (after finding the best parameters for the different data sets) and (2) our SRVM algorithm for three different data sets of varying class and instance number. Generally, the accuracies for both methods are comparable. The virtue of the SRVM method (apart from being systematically able to detect optimal parameters by examining the inter-replica overlap) is that the SRVM suffers from far less data bias that SVM; this will be made later in the text and in Table 3.4.

model becomes more sophisticated and the prediction accuracy rises rapidly. This shows that the model can be quite accurate even with a low number of fixed vectors. After a certain point, increasing the number of fixed vectors v starts leading to over-fitting and the prediction accuracy drops, however the drop is rather gradual, indicating that the model is robust against overfitting.

The dependence of the accuracy on the replica number $\mathcal{R}$ was tested in the Australian dataset by performing 50 five-fold CVs and taking the average accuracies across the SRVM results with $v = 50$ anchor points for the Gaussian kernel and investigating the results when the number of replicas $\mathcal{R}$ was varied from 1 to 89. The results are plotted in Fig. 3.11(b).

In addition to assessing the accuracy of the SRVM algorithm, it is important to compare its performance to established learning algorithms and to try and quantify any relative advantages and/or deficiencies. To that end, as we noted earlier, we took the Support Vector Machines (SVM) algorithm [31, 32] as a baseline for comparison. For the LSVT dataset, we used a 'brute force' method of finding the optimal parameters for this contender to our method- the SVM model- by running it for all values in a grid in parameter space. Once the optimal parameters were found, it was observed that the maximum accuracy for SVM was 0.873. The difference in accuracy between

our SRVM method (in which optimized parameters were found by replica overlap not by comparing to the solution) and the standard SVM algorithm (now optimized to achieve highest accuracy) is 0.016. This difference is not statistically significant, so the relative advantage of either method might not be immediately clear.

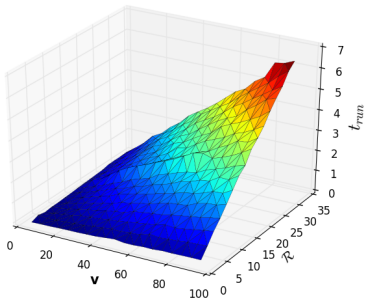
To dig further into the comparison, while simultaneously exploring the SRVM performance on a deeper level, we next examine the runtime of both algorithms. We ran the SRVM algorithm on the LSVT dataset for various values of v and $\mathcal{R}$. The runtime is considered to be the time that it takes to calculate the average CV accuracy, and does not include finding the optimal number of parameters, or preprocessing steps. Figure (3.12a) shows a 3D surface plot of the runtime versus the number of anchor points and replicas. In figures (3.12b) and (3.12c), the runtime is exhibited as a function of the number of anchor points v and the number of replicas $\mathcal{R}$. The data make clear that the runtime increases linearly with increasing v and $\mathcal{R}$. This observation suggests that it is possible to find the runtime at low numbers of both variables in order to assess how long a run will take with larger values.

The general optimization of model performance involves maximizing accuracy while simultaneously minimizing the necessary runtime. Therefore, it would be beneficial to have a measure of the compounding of these two goals. To assess the intersection of accuracy and run time, we can define a metric which we call the coefficient of performance, τ, which we define as

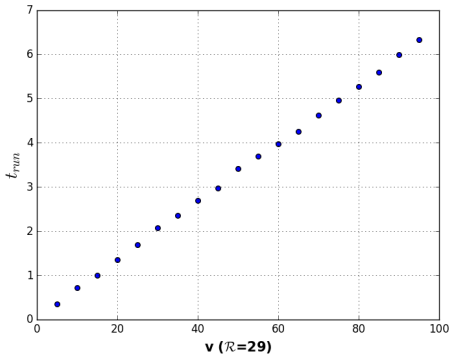
$$\tau \equiv \frac{\bar{A}}{t_{run}}. \tag{3.8}$$

This metric allows for an efficient via for simultaneously looking at optimal accuracy and run time. Using the results of the runtime and accuracy measures discussed above for the LSVT dataset, we calculated the values of τ as a function of v and $\mathcal{R}$. Detailed results highlighting various aspects are shown in Figs. (3.13a,3.13b,3.13c).

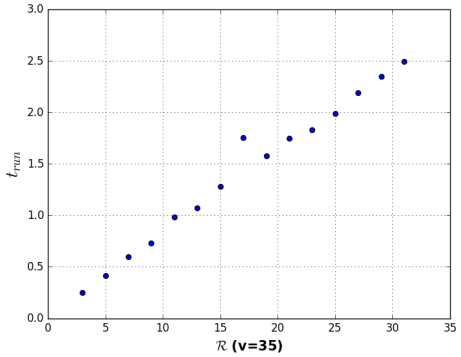
It is clear that the COP decays as a function of both the number of anchor points v and the number of replicas $\mathcal{R}$. This is consistent with the linearity of the runtime and the asymptotic behavior of the accuracy. Locating the 'knee' in the COP data,



(a)

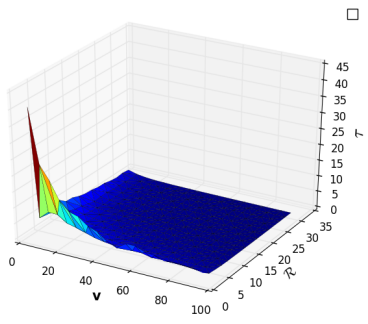


(b)

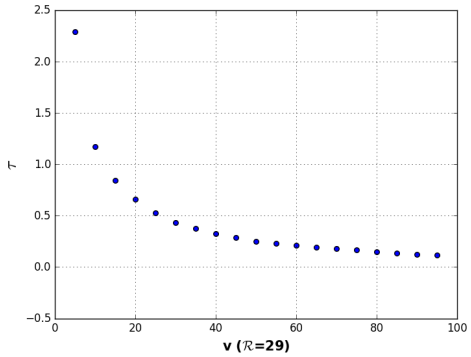


(c)

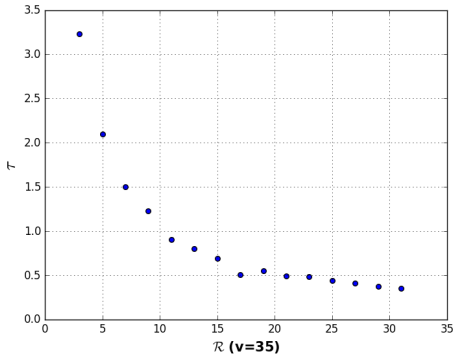
Figure 3.12: (Color Online.) LSVT data set. (a) 3D surface plot of SRVM runtime as a function of the number of anchor points v and number of replicas, $\mathcal{R}$. (b) Projection of runtime surface of (a) into the accuracy-anchor point plane with constant replica number, $\mathcal{R}$=29 to show runtime as a function of number of anchor points. (c) Projection of runtime surface of (a) into the accuracy-replica plane with constant number of anchor points, v=35 to show runtime as a function of the number of replicas. It is clear that in both cases, the runtime scales linearly with both parameters, allowing for prediction of runtime from from small parameter samples. If it known that many coefficients (a single coefficient is associated with each anchor point) will be needed in Eq. (3.1) then one may estimate the requisite runtime of anchor points from the known runtime from smaller v.



(a)



(b)



(c)

Figure 3.13: (Color Online.) Analysis of the LSVT data set. (a) 3D surface plot of SRVM Coefficient of Performance (COP) of Eq. (3.8) as a function of the number of anchor points v and number of replicas $\mathcal{R}$. (b) Projection of COP surface of (a) into the accuracy-anchor point plane with constant replica number, $\mathcal{R}$=29 to show COP as a function of number of anchor points. (c) Projection of COP surface of (a) into the accuracy-replica plane with constant number of anchor points, v=35 to show COP as a function of the number of replicas. As they trivially must, the trends for the COP of Eq. (3.8) in all panels encapsulate the behavior of both the accuracy (Fig. (3.9) and (near linear) run time (Fig. (3.12)) dependence on v and $\mathcal{R}$.

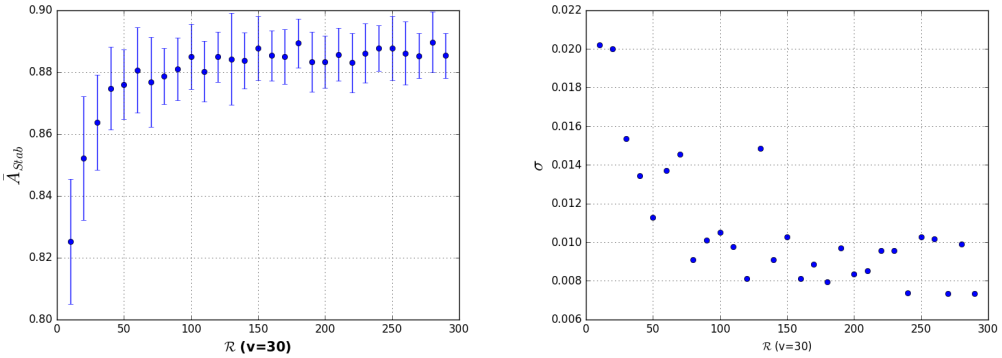


Figure 3.14: (Color Online.) LSVT data set. A defining feature of the SRVM is that each replica is stochastically generated (leading, a priori, to different results.) To that add end, (a) we display the average accuracy with $v = 30$ anchor points for variable numbers of replicas, simulated 20 times each with different random replica generation seeds in each simulation. Associated standard deviations are shown as error bars. (b) Plot of these standard deviations in the accuracy that are associated with runs for various replica numbers. The monotonic decrease in the standard deviation with increasing number of replicas demonstrates that prediction results become more stable with increasing replica number $\mathcal{R}$; when additional replicas (for an increasing yet still small $\mathcal{R}$) vote, the final outcome becomes progressively more stable to statistical fluctuations from the stochastic generation of the anchor point vectors. The plot further makes clear that the standard deviation quickly reaches a leveling-off point at which further replica increase does not have a statistically significant impact on stability.

allows for extracting reasonable values of the parameters for the trade-off between accuracy and runtime.

In addition to calculating COPs for the SRVM algorithm, we also calculated them for SVM. Broadly, the SVM COP is considerably better than that of SRVM, and this is entirely due to the fact that SVM runs much quicker. This is likely due to the fact that the SVM algorithm has been highly streamlined and optimized in various software packages over the decades, whereas our algorithm is new. It could also be due to the fact that for all our implementations of our algorithm we computed the pseudo inverse exactly (Eq. (3.3)) instead of approximating it with methods such as gradient descent. Further massaging of the algorithm structure will likely decrease the runtime.

3.4.2 Stability

A central characteristic of the SRVM algorithm is the use of voting between replicas to increase the accuracy. Intuitively, one would expect that the number of replicas and the accuracy should be positively correlated: more replicas give better accuracy. Another quintessential feature of the SRVM is that the v anchor vectors associated with each individual replica are generated stochastically. This allows for a robust classification of new instances. This also means that each run of the algorithm will be different, with different outcomes possible. Therefore, it is important to examine the stability of the output. It is expected that for a low number of replicas ($\mathcal{R}$), the overall vote can change rather dramatically with different runs, so the accuracy can fluctuate. It is further expected that as the number of replicas increases, the fluctuations will be suppressed by the presence of more information in the overall vote. To test this, we ran the SRVM algorithm on the LSVT dataset with v=30 anchor vectors per replica 20 times each, for varying number of replicas. In panel (a) of Fig. (3.14), we display the average accuracy across all 20 runs with a fixed number of replicas as this number ($\mathcal{R}$) increases. The error bars in the figure reflect the standard deviation in accuracy. In panel (b) of Fig. (3.14), we plot the standard deviation in accuracy versus number of replicas. From the panels in Fig. (3.14), it is clear that the standard deviation decreases rapidly with increasing number of replicas and eventually levels off to a roughly constant value. This is consistent with the earlier observation that the runtime is linear and the accuracy approaches a leveling off before decreasing. Further, the result implies that beyond a certain number of replicas, the overall accuracy is largely stable to fluctuations associated with stochastic generation of anchor vectors, thus alleviating a potential weakness of the method.

Tie stability was further tested using the Australian dataset by performing 50 five-fold CVs and computing the average accuracies across models with replica numbers ranging from 1 to 89. The results are provided in Fig. 3.11. As this figure makes evident, for this dataset, the average accuracy rises relatively quickly at the beginning from just one replica and maintains a general monotonic trend as the replica number

increases. We begin to observe diminishing returns somewhere after 15 replicas. This is to be expected, as the amount of available information in the dataset is objectively limited so there is a cap on achievable accuracy. Note that since we do a simple majority vote, the replica numbers are all odd to ensure that no ties appear during voting.

One would expect that as the number v of fixed vectors increases, initially the fitted model becomes more sophisticated and the prediction accuracy rises. After a certain point increasing the number of fixed vectors starts leading to over-fitting and the prediction accuracy drops. Using more fixed vectors also results in a slower algorithm. Therefore it would be very useful if we had a way of estimating how many fixed vectors are appropriate for a certain problem.

In Fig. 3.11(a), we notice that the curves for both the average accuracy and the average replica overlap rise rapidly from their values for a single fixed vector ($v = 1$) to a nearly flat maximum that appears when the number of around fixed vectors $20 \lesssim v \lesssim 100$; when $v \gtrsim 100$, the accuracy begins to taper off due to the alluded to overfitting. The two curves indeed follow each other closely, supporting the notion of using replica overlap to estimate the dependence of the expected accuracy on v. We found similar behaviors for other parameters other than the anchor vector number v.

There are some other outstanding features of the figure: the rapid rise of the two curves at low fixed vector number shows that the model can be quite accurate even with low fixed vector number, and the slow tapering off of the two curves indicates that the model is robust against overfitting.

3.4.3 Impact of pre-processing

In the beginning of this section, we alluded to the possibility that the specific pre-processing method employed may have an impact on the performance of the SRVM algorithm. In this subsection, we will examine the impact of preprocessing the data using feature scaling on our final results. We feature scaled in three different ways:

• (1) linearly transforming the data such that domain of each feature over the entire data set ranges from 0 to 1

• (2) linearly transforming the data such that each feature assumes values in $[-1, +1]$, and

• (3) normalizing the scaled data with mean 0 and standard deviation equal to unity.

We examined the average accuracies (and their variances) associated with these three different preprocessing methods using statistical tests. The results (see table 3.2) demonstrate that one must absolutely reject the null hypothesis H_0 that all of the means are equal. The disparate preprocessing methods definitely lead to different results. The specific testing of the means were performed both (i) assuming normal distribution of the averages (the f-statistic) and (ii) without this assumption (the h-statistic using Kruskall Wallis test [35]). Both tests revealed that the average accuracy was not statistically uniform across all methods of feature scaling. To investigate which methods were different, individual t-tests of the means were undertaken. We performed three different t-tests [36]. These tests demonstrate that there is no difference between pre-processings to the types (1,2) (the 0 to 1 and (-1) to 1 scalings). However, these two cases however are different from the normalization (preprocessing type (3)). The normalization preprocessing tends to be the most accurate of the three for lower numbers of anchor points (v) and replicas ($\mathcal{R}$). In general, the datasets that were scaled normally (preprocessing (3)) had higher accuracy with lower numbers of anchor points. We further tested that the variances were equivalent using a Levene test [37]; the results indicated no statistically significant difference between the variances across all preprocessing methods employed. Overall, these outcomes demonstrate that one must consider the specific type of preprocessing undertaken, when assessing the performance of the SRVM algorithm.

Null Hypothesis	Test Statistic	p-Value	Conclusion
$\mu_1=\mu_2=\mu_3$	f=4.4542	p=0.0159	Reject H_0
$\tilde{x}_1=\tilde{x}_2=\tilde{x}_3$	h=8.9006	p=0.0116	Reject H_0
$\sigma_1^2=\sigma_2^2=\sigma_3^2$	w=2.1240	p=0.1289	Fail to Reject H_0
$\mu_1=\mu_2$	t=2.7440	p=0.0092	Reject H_0
$\mu_1=\mu_3$	t=2.7440	p=0.0092	Reject H_0
$\mu_2=\mu_3$	t=0	p=1.0	Fail to Reject H_0

Table 3.2: Results of statistical hypothesis tests undertaken to assess whether different pre-processing techniques impact algorithm accuracy for the same sets of parameters (v and $\mathcal{R}$)

3.4.4 Optimization via replica overlap metrics

When choosing the optimal values of the parameters for a learning algorithm, it is helpful to have a reference function which does not require the calculation of the accuracy, which still relays information about model performance. This gives a more 'fair' way of choosing the best values of the parameters without a brute force method. In the SRVM algorithm, because we have many replicas which are voting together by a simple majority vote, it seems reasonable that some measure of the overlap between the replicas would be a measure of model performance. Indeed, when all of the replicas are largely in agreement, it should imply the model is performing optimally and vice versa. However, it is possible that all of the replicas could be in agreement, with all of them being incorrect. Therefore, it is important to test whether proposed replica overlaps are agreement with the accuracy. To test this, we propose two different replica overlap functions, and test them on the LSVT dataset. The first overlap function is defined as

$$\mathcal{O}_1 = \sum_{\alpha>\beta} \vec{y}^{\,\alpha} \cdot \vec{y}^{\,\beta} \tag{3.9}$$

and measures the total overlap of the predicted labels of all replicas. For each of the replicas $1 \leq \alpha \leq \mathcal{R}$, the vectors $\vec{y}^{\,\alpha}$ have g components (where g is the number of distinct predicted (or, in some rare cases, fitted) data points $\vec{x}$ in each replica α). This metric may be trivially averaged by dividing by the total number of distinct replica

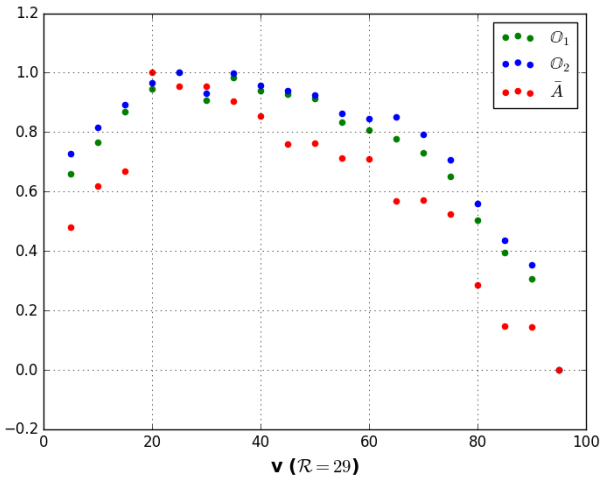


Figure 3.15: (Color Online.) LSVT data set. Plot of the two overlaps O_1 and O_2 (each of which is now scaled by their respective maximum value) of accuracy and the accuracy as a function number of anchor points v for $\mathcal{R}$=29 replicas. It is observed that both overlap (associated with almost identical numerical values) scale with the accuracy of the predictions. As in the other examples that we studied, this correlation (and others like it) illustrates that instead of having to rely on exact calculations of the accuracy one may use the overlaps to ascertain the optimal values of the parameters defining the SRVM model (in this case, the optimal number of anchor points v and the number of replicas $\mathcal{R}$).

pairs, i.e., by multiplying the righthand side of Eq. (3.9) by $\frac{2}{g\mathcal{R}(\mathcal{R}-1)}$. A similar, but computationally more efficient overlap function is defined as

$$\mathcal{O}_2 = \sum_{\alpha} \vec{y}^{\,\alpha} \cdot \vec{\mathcal{V}} \tag{3.10}$$

which measures the overlap of each replica with the overall vote as determined by Eq. (3.6). The calculation of Eq. (3.10) requires storing much less information the overlap defined in Eq. (3.9). Similar to each vector in the set $\{\vec{y}^{\,\alpha}\}_{\alpha=1}^{\mathcal{R}}$, the vector $\vec{\mathcal{V}}$ has g components (one component (the voted prediction) for each of the g distinct examined data points $\vec{x}$). (Similar to Eq. (3.9), an average is trivially calculated by dividing the righthand side of Eq. (3.10) by $(g\mathcal{R})$.) In Fig. 3.4.4, we plot the results of the overlap measures of the LSVT dataset. In these panels, the number of anchor points, v, is plotted on the x-axis with datasets for increasing values of replica numbers, $\mathcal{R}$, with increasing $\mathcal{R}$ being proportional to increasing y-axis values, shown. The data in Fig. 3.4.4 seems to display the same overall characteristics as that of the accuracy shown earlier, but it is important we compare them directly. In Fig. 3.15, we plot the average accuracy, and two overlap functions for $\mathcal{R}$=29 replicas and various numbers of anchor points, all scaled by their maximum values so as to be able to fall on the same plot. It is abundantly clear from the data, that the two definitions of the replica overlap and the accuracy scale in a one-to-one fashion, making either overlap function an excellent candidate to find the optimal parameter values without necessitating the calculation of model accuracy. Similar trends are seen in Figs. (3.10(b), 3.16,3.17, 3.18, 3.19).

In many cases, having some measure of the probability of a point being labeled correctly is important for identifying the location of classification boundaries. Again, the existence of SRVM replicas allows for getting a measure of these probabilities. We can define a third single data point overlap across replicas which can approximate the probabilities of classification. This allows us to determine whether a given point $\vec{x}_i$ is near a classification boundary, as when result in different labels, the point is likely near a boundary. We call this single instance overlap the 'Agreement', and define it

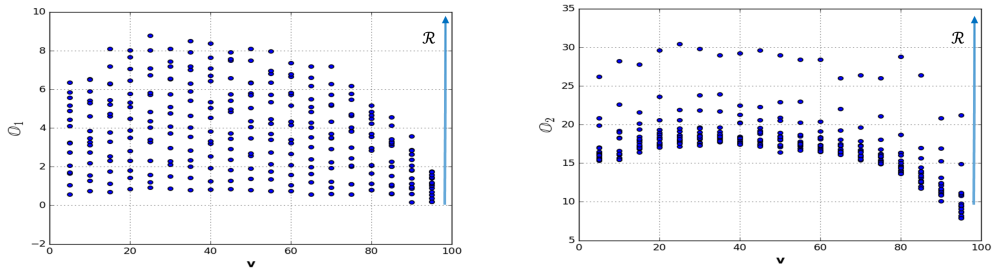


Figure 3.16: (Color Online.) LSVT data set. Plot of the overlap functions defined in Eqs. (3.9,3.10) as a function of the number of anchor points v for various numbers of fixed replica numbers. The replica number increases along the vertical axis.

as

$$\mathcal{A}_i \equiv \frac{1}{\mathcal{R}} |\sum_{\alpha=1}^{\mathcal{R}} y_i^\alpha|. \tag{3.11}$$

This single instance metric complements the global overlaps of Eqs. (3.9, 3.10). In general, the overlap $\mathcal{A}_i$ increases in tandem with the probability $\mathcal{P}_i$ of correctly classifying a given point $\vec{x}_i$.

The bar chart in figure 3.20 (a) shows how well the predictions of different replicas agree. The majority of points fall in the last bin (meaning that for most points, different replicas predicted the same result). We also calculated the corresponding accuracy for each bin in Fig. 3.20(b). Again it is apparent from the bar chart that the points with the maximum replica voting agreement have the highest accuracy as it is expected. As mentioned earlier, the Heart data consists of 270 data points each of 13 features. Replicas are of 50 points in thirteen dimensions. The average accuracy after 10 runs of voting is 81.25%. The corresponding accuracy of SVM is 82.4%. We see our prediction algorithm is acting very close to SVM.

The correlation between availability and classification accuracy was also investigated using the Australian dataset. The Australian data set was used to demonstrate the correlation between replica overlap and datapoint prediction accuracy. For this test, we used Gaussian kernel models containing 31 replica with 50 anchor points each. A total of 10 models were generated, and a 5-fold cross-validation on the data

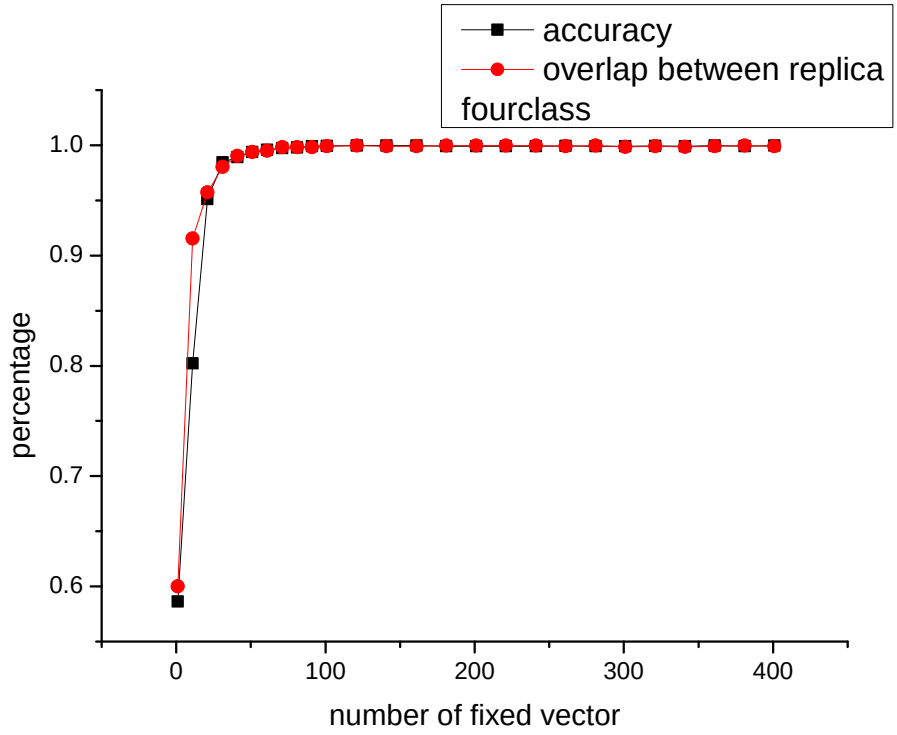


Figure 3.17: A comparison between (1) the average replica overlap (as computed via the averaged variant of Eq. (3.10)) and (2) average accuracy of a model with $\mathcal{R} = 5$ replicas for the Four-class benchmark when using the SRVM algorithm with a Gaussian kernel. The horizontal axis corresponds to the number of anchor points v.

set was ran on each model. For each 5-fold cross-validation, every data point in the data set will be a test data point exactly once. Any test data point x_i was classified by its replica agreement $\mathcal{A}_i$. In Fig. 3.21, we binned the test data points based on their agreement values (multiplied by replica number ($\mathcal{R} = 31$) for clarity of presentation) and calculated an aggregate accuracy for each bin. We see that indeed the data points with higher replica agreement in general are also being predicted with higher accuracy, showing a clear positive correlation between replica agreement and prediction accuracy. Another feature from Figs. (3.11, 3.25) is that the vast majority of data points have good replica agreement. In Figs. (3.22, 3.23, 3.24), we report on similar tendencies found for the Four-class, Svmguide1, and liver disorder benchmarks.

The Australian dataset was also used to show that the average agreement across the dataset is also a useful replica overlap function. Using the same procedure for Fig. 3.11, we used Gaussian kernel models with varying number of anchor points

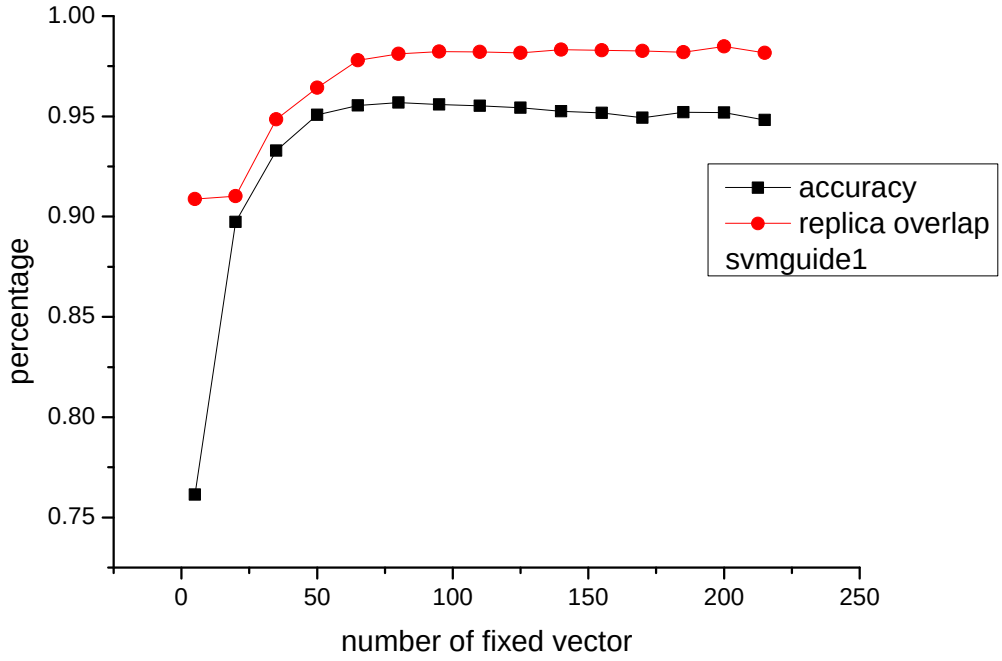


Figure 3.18: Plotted on the same axes are (1) the average overlap between different pairs of replicas (calculated with the replica averaged variant of Eq. (3.10)) and (2) the average accuracy as a function of the number of anchor points v. This analysis was performed for the "Svmguide1" benchmark classification problem with the Gaussian based SRVM algorithm with $\mathcal{R} = 5$ replicas.

and 5 replicas each, and calculated the average agreement of all datapoints. The results are plotted with the average accuracy in Fig. 3.25(a), and we see strong correlation between the average accuracy and the average agreement. If we introduce the average RMS error as the learning free energy $\mathcal{F}$ scaled by the number of data points: $RMS = \mathcal{F}/N$, we see in Fig. 3.25(b) the average RMS error correlates negatively with both the average accuracy and the average agreement.

In Fig. 3.26, we show the correlation between averaged replica overlap and averaged accuracy for different number of anchor points. The range of anchor points runs from 10 to 500 while the number of replicas are being kept fixed at 31. Similar to the Australian and other data sets that we analyzed, both the replica overlap and the accuracy closed tracked one another (and further correlated with the value of the energy function of Eq. (3.4)). Here, these quantities were non-monotonic as a func-

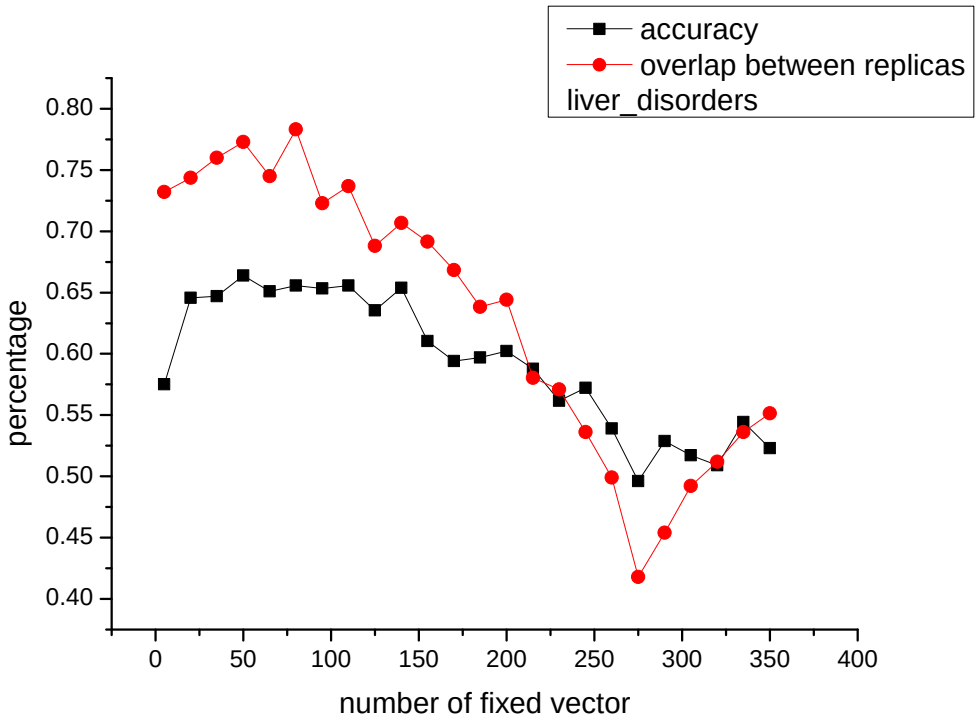


Figure 3.19: The results of the SRVM algorithm for a Gaussian kernel with $\mathcal{R} = 5$ replicas for the "liver disorder" dataset. Similar to Figs. (3.17,3.18), we plot the inter-replica overlap and accuracy as a function of the number of anchor points v on the same set of axes.

tion of the number of anchor points. The "energy" curve is this figure corresponds to the average of Eq. (3.4) over 31 different replica realizations. Five-fold CV was employed in our tests for the accuracy of the predictions. Perusing this Figure, we see that the averaged penalty function of Eq. (3.4) becomes minimal when the highest inter-replica overlap is achieved and when the predicted classifications are of the highest accuracy.

In Figs. (3.27, 3.28, 3.29), we similarly demonstrate the correlation between the found accuracy and the lowest value of the penalty function (or energy) of Eq. (3.4) for different data sets.

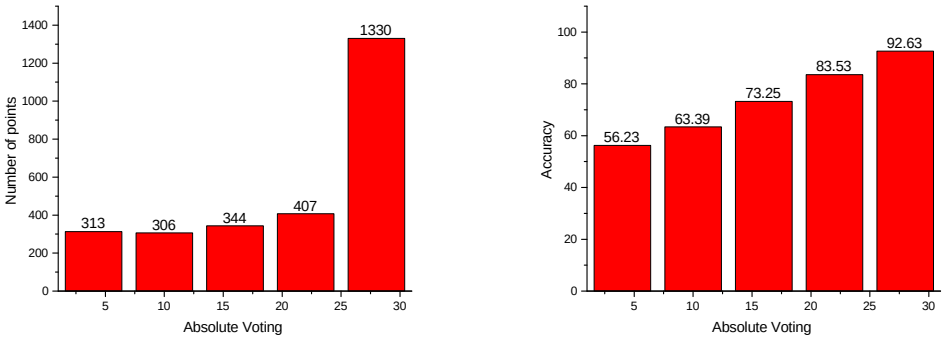


Figure 3.20: (Color Online.) An analysis of the Heart benchmark via $\mathcal{R} = 31$ replicas. The Heart benchmark has 270 data points. The CV calculations were replicated 10 times so, overall, $270 \times 10 = 2700$ points were classified. (a) Bar chart showing the distribution of agreement values of the Heart benchmark data points across 10 five-fold cross-validations. The rightmost bar denotes the number of points (1330 out of 2700) that were in nearly the same way by all 31 replicas. The leftmost bar corresponds to the 313/2700 data points that were classified with a minimal Agreement (Eq. (3.11)) amongst the 31 replicas. (b) A histogram of the average accuracy of the Heart benchmark data points in each bin of the Agreement values. Higher Agreement positively correlates with a higher average accuracy.

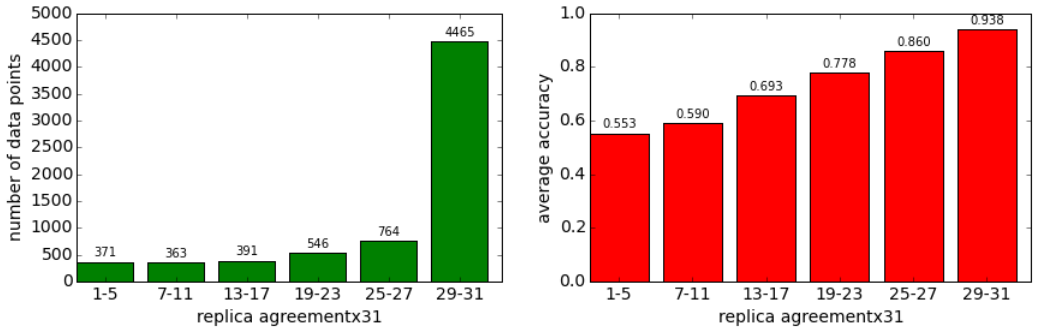


Figure 3.21: (Color Online.) "Australian" data set. (a) Bar chart showing the distribution of agreement values of the data points across 10 five-fold cross-validations. (b) Bar chart showing average accuracy of the data points in each bin of agreement values. Higher agreement positively correlates with higher average accuracy. On the horizontal axis, we plot the un-normalized sum of Eq. (3.11), i.e., $|\sum_{\alpha=1}^{\mathcal{R}} y_i^\alpha|$.

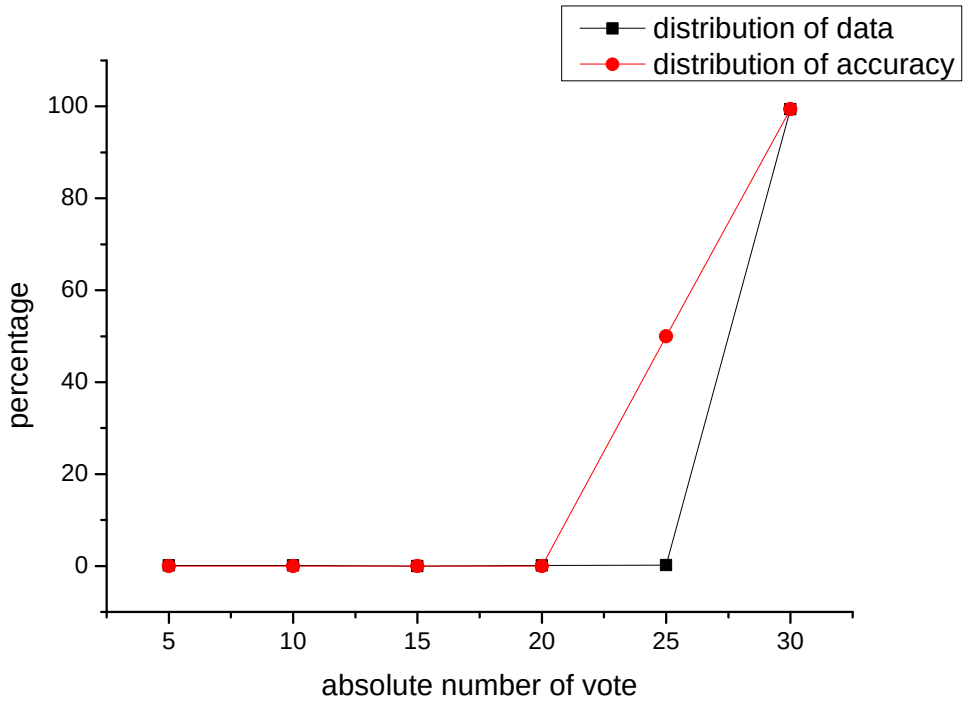


Figure 3.22: (Color online.) Distribution of data points and accuracy for the Four-class benchmark. On the horizontal axis, we plot the"absolute number of votes"- the un-normalized sum of Eq. (3.11), i.e., $|\sum_{\alpha=1}^{\mathcal{R}} y_i^\alpha|$. The "distribution of data points" marks which fraction of the data points have a given "absolute number of votes" (thus the sum of this distribution over all possible "absolute number of votes" is unity). The accuracy curve is, generally, monotonic in the replica overlap as is manifest here by the "distribution of data points" fraction.

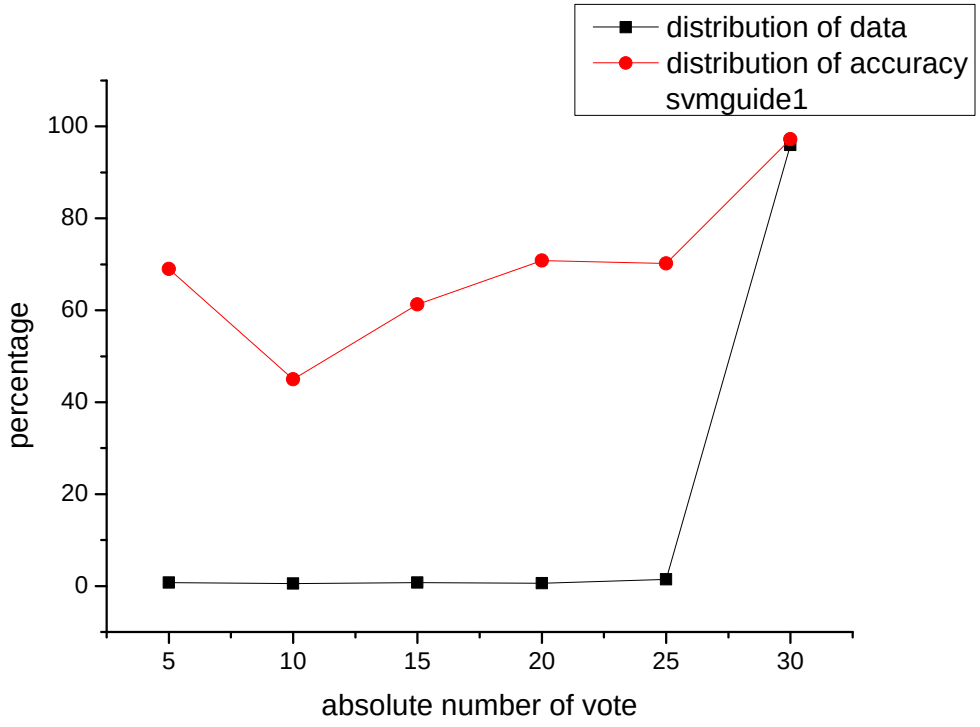


Figure 3.23: (Color online.) The distribution of data points and accuracy for the Svmguide1 dataset. See the caption of Fig. 3.22 for the definition of the axes and curves.

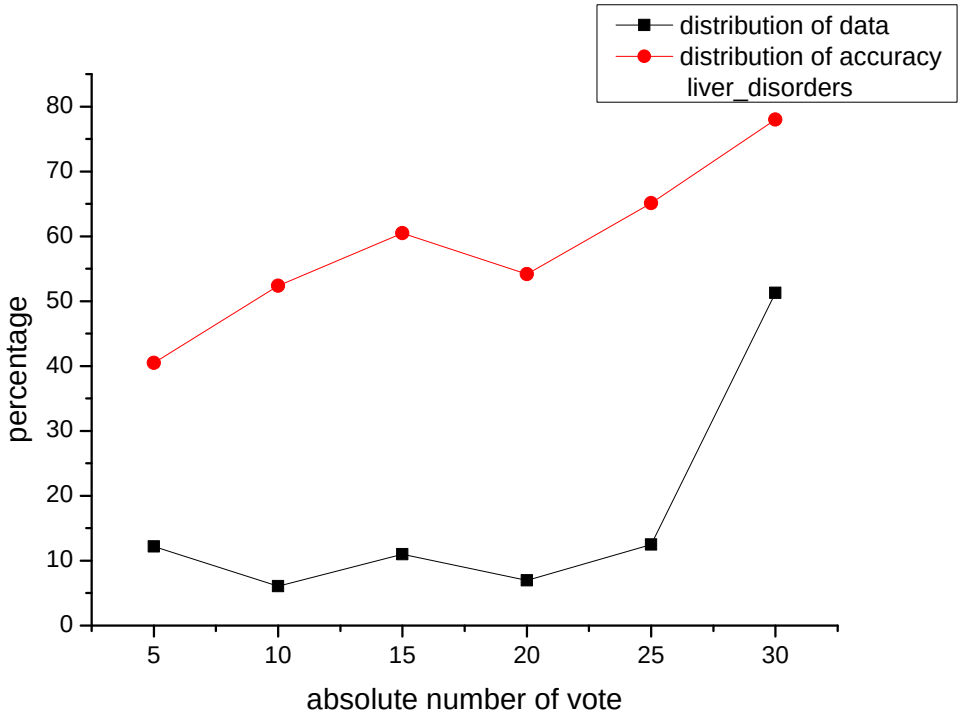


Figure 3.24: (Color online.) The correlation between the replica correlations ("distribution of data points") and accuracy for the liver disorder benchmark. The definition of graph is similar to that in the caption of Fig. 3.22.

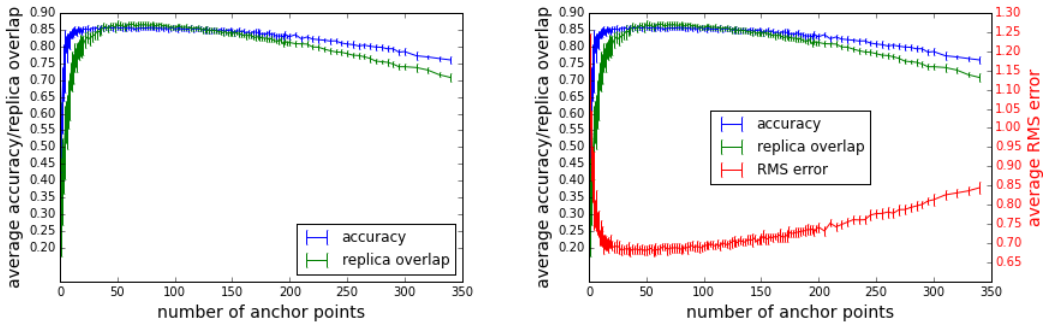


Figure 3.25: (Color Online.) Australian data set. Each data point is the average over 20 runs. (a) Average agreement and average accuracy with varying number of anchor points (similar to Fig. 3.11). (b) Average agreement, average accuracy and average RMS error with varying number of anchor points.

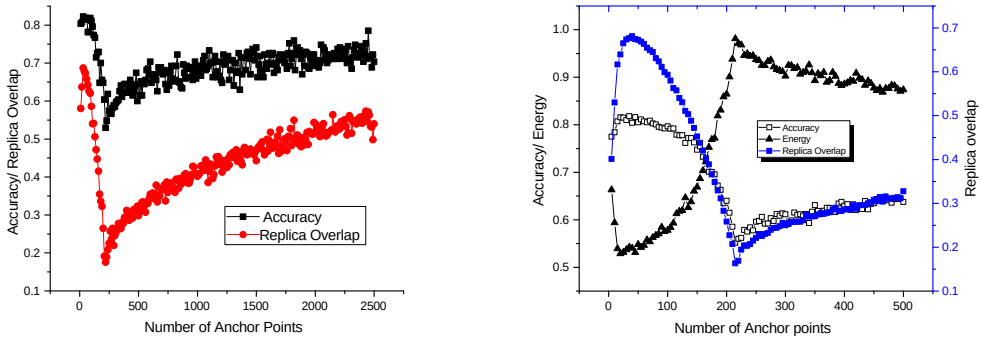


Figure 3.26: (Color Online.) Heart data set. Each data point is average of 20 runs. (a) Average agreement and average accuracy with varying number of anchor points. (b) Average agreement, average accuracy and average RMS error with varying number of anchor points.

3.4.5 Class imbalance and alternative performance metrics

In Fig. (3.30), we depict the results of a principal component analysis (outlining the two dominant principal components). This analysis enables us to visualize where our algorithm fails to find the correct answer for the LSVT data set. As seen in this figure, while there is no apparent distribution in principal component space of the cases that we obtained incorrectly, the two classes are massively imbalanced (as is often the case in classification sets). Other metrics are necessary to compare our results to those of SVM (and other algorithms). To that end, we briefly regress to the "accuracy paradox" [38]. This colloquial "paradox" is simple to explain: if the data set given is heavily imbalanced so that most of the provided data belong to one type, one might as well just guess the dominant answer every time and miss subtle instances. This must be taken into consideration. To that end, in Fig. (3.31), we provide a confusion table and look at how well we perform while qualifying true positives and negatives and false positive and negatives. The specificity and sensitivity are related to true positives and false negative rates. This information may be used to compute various metrics; these measures combine class imbalance, specificity, sensitivity, and performance into one number. Notable metrics in Table 3.4 are those reflecting Cohen's κ. In terms of these κ values. the superiority of SRVM over SVM is made apparent (by a very large statistically significant difference). The statistical measures indicate

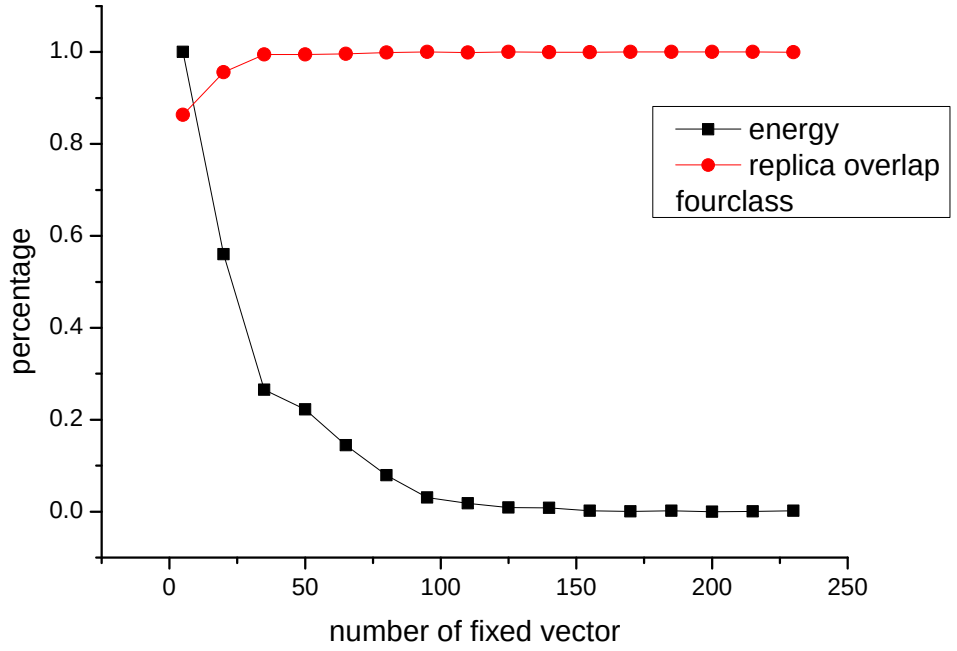


Figure 3.27: A comparison of the average replica overlap and average energy of a model with 5 replicas for the Four-class data set.

that SVM does a lot more "guessing" that the SRVM algorithm and tends to become "lucky" by predicting the dominant class more often. This is to be expected since SVM segments feature space with a specific kernel at a hard cut and only considers points on a specific boundary that need to be carefully classified. By contrast, the distinguishing attribute of SRVM is that numerous stochastic replicas are considered- a characteristic that tends to lead to less bias. Taken together, the Cohen's κ values [39], along with the F_1 [40] (that does not take into account true negatives) and C_M (Matthews Correlation coefficients) [41] metrics illustrate that SRVM exhibits a statistically significant advantage over the SVM algorithm insofar as the lack of inbuilt class imbalance bias is concerned.

3.4.6 Layered voting: multiple kernels and recursive learning

We can add hidden layers to the SRVM by allowing different kernels to all vote. Each kernel predicts an outcome on its own for each instance. We may combine voting

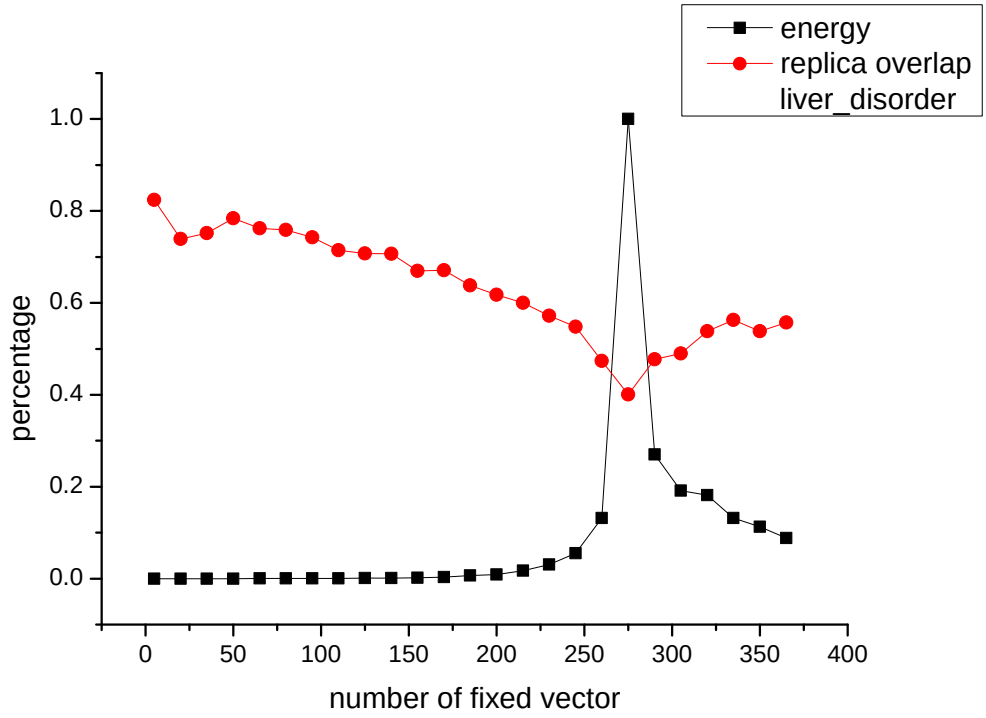


Figure 3.28: A comparison of the average replica overlap and average energy of a model with 5 replicas for the liver disorders data set.

results from different kernels to come together by voting anew from the results from the first (single kernel) votes, see Fig. (3.32). The advantage of this modus operandi is that we can adjust weights for the different functions. Without adjusting for weights, instead of Eq. (3.6), one may use the more general average of

$$\mathcal{V}_i = \frac{1}{N_k \mathcal{R}} \sum_{\alpha=1}^{\mathcal{R}} \sum_{k=1}^{N_f} y_{i,k,p}^{\alpha}, \tag{3.12}$$

where the predicted value $y_{i,k,p}^{\alpha}$ for replica α is found using Eq. (3.1) with kernel K belonging to the $k-$th entry of the list of Eq. (3.2) or other trivial extensions thereof (and N_f the total number of functions in such lists). The weight of one function may be adjusted as the calculation proceeds to be higher or lower to increase the accuracy. Without adjusting for weights, we see in Figs. (3.33,3.34,3.35) that the accuracy, run time, and coefficient of performance are similar those that we obtained earlier (within a single layer voting model- the usual SRVM). A trivial extension of Eq. (3.12) is

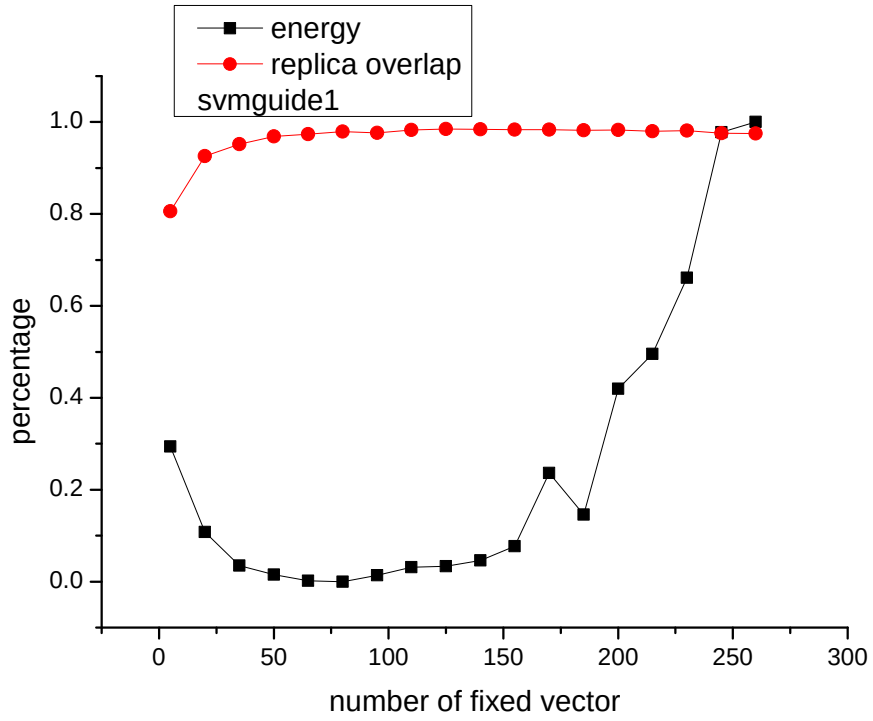


Figure 3.29: A comparison of the average replica overlap and average energy of a model with 5 replicas for the Svmguide1 benchmark data set.

that of the weight adjusted voting,

$$\mathcal{V}_i = \frac{1}{\mathcal{R}} \sum_{\alpha=1}^{\mathcal{R}} \sum_{k=1}^{N_k} w_k y_{i,k,p}^{\alpha}, \tag{3.13}$$

with the weights w_k satisfying, $\sum_{k=1}^{N_f} w_k = 1$.

As we explained in Section 3.4.4, one may aim to find the optimal parameters by noting when these lead to a maximal overlap between the replicas. Additionally, of course, one may see when these lead to accurate solutions- yet that either requires "cheating"- i.e., (1) adjusting the parameters to obtain the known answer or to (2) the removal of some of the known input data to use it as a CV test (the latter case is non-optimal since already known data are removed from the training set). At any rate, testing for overlaps and/or direct accuracies by brute force change of parameters can be taxing. An alternate approach for *determining the optimal parameters and weights* such as those of w_k in Eq. (3.13) (and trivial multi-layer generalizations thereof),

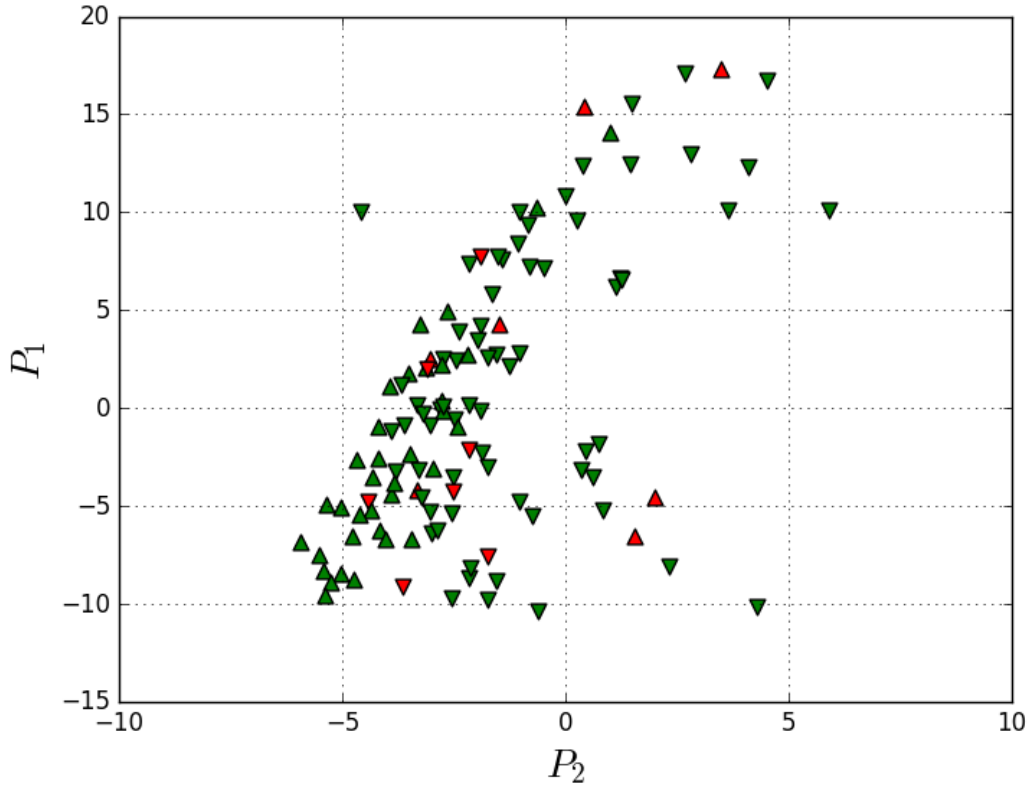


Figure 3.30: (Color Online). The LSVT data set projected into the plane of the first two principle components, so as to visualize model performance. Data points denoted with an upward pointing triangle are points with known label '+1' and downward pointing triangle are those points with known label '-1'. Points which are colored green correspond to points correctly classified by the SRVM algorithm, whereas points colored red, were incorrectly classified.

	SRVM-Gauss			SVM		
Fold 1		Predicted		**Fold 1**		Predicted
		+1	-1			+1	-1
Actual	+1	8	3	Actual	+1	6	5
	-1	2	13		-1	1	14
Fold 3		Predicted		**Fold 3**		Predicted
		+1	-1			+1	-1
Actual	+1	6	2	Actual	+1	5	3
	-1	0	17		-1	0	17
Fold 5		Predicted		**Fold 5**		Predicted
		+1	-1			+1	-1
Actual	+1	6	1	Actual	+1	6	1
	-1	1	17		-1	2	16

Figure 3.31: LSVT data set. Confusion Tables constructed from three folds of a five-fold cross validation for the SRVM and SVM algorithms. Overall, the confusion tables make clear that the SRVM algorithm is slightly less inclined toward false negatives (FN) than SVM, which is and important result, as the class imbalance is toward the negative side in the dataset.

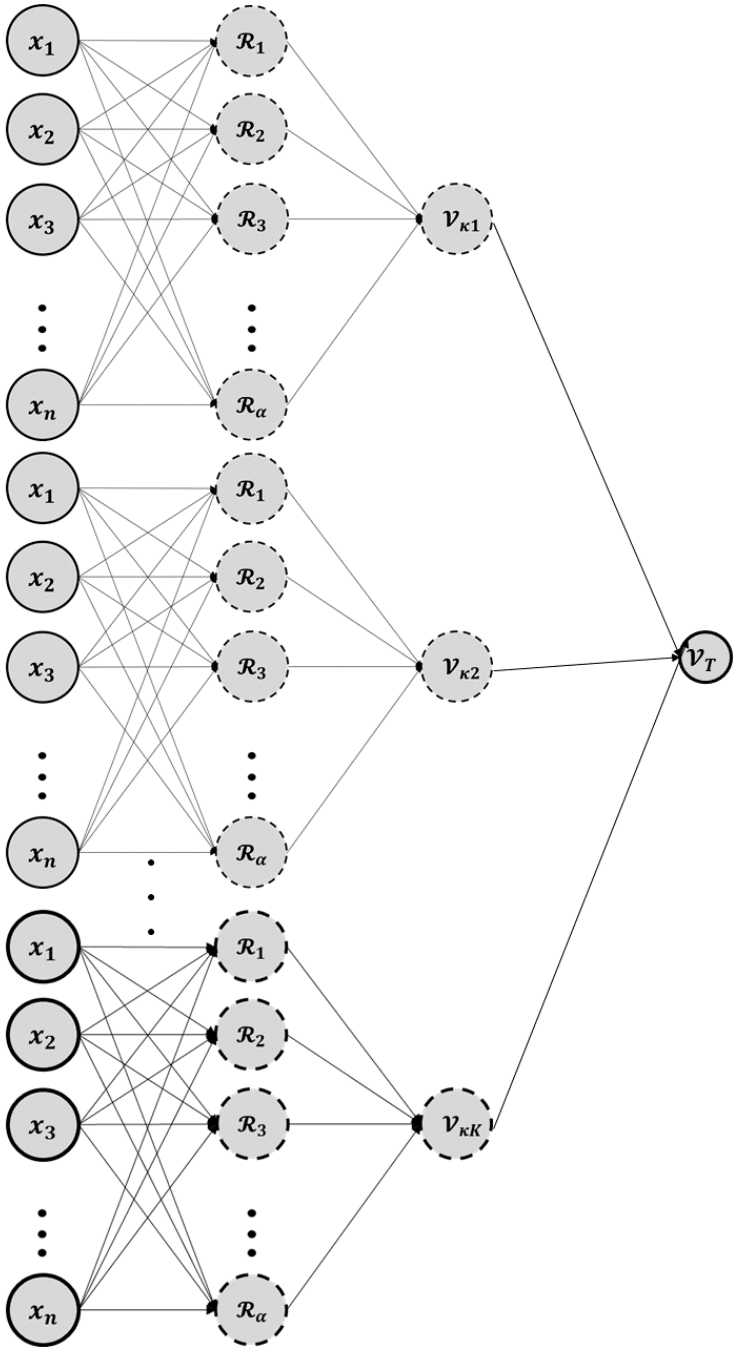


Figure 3.32: (Color Online.) Schematic representation of the layered kernel approach to the SRVM method. By allowing multiple kernel functions to each vote, after themselves being obtained via replica voting, adds a hidden layer such that the SRVM algorithm acts like a traditional neural net. This allows for adding weights to the given kernels as they vote to increase accuracy. This will be taken up in a future paper.

Label	Fold 1	Fold 3	Fold 5	Average
SRVM Right, SVM Right	18	20	22	20
SRVM Wrong, SVM Wrong	4	2	2	2.67
SRVM Right, SVM Wrong	2	1	0	1
SRVM Wrong, SVM Right	1	0	0	0.33
SRVM Mixed, SVM Right	1	2	0	1
SRVM Mixed, SVM Wrong	0	0	1	0.33

Table 3.3: Number of testing data points corresponding to a given pair of outcomes for the Gaussian-Kernel SRVM and SVM algorithms across three different folds of a cross-validation set.

Fold	*Classifier*	*Sensitivity*	*Specificity*	*Precision*	κ	F_1	C_M
21em**1**	SRVM-Gauss	0.727	0.866	0.8	0.56	0.761	0.603
	SVM	0.545	0.933	0.857	0.436	0.667	0.533
21em**3**	SRVM-Gauss	0.75	1	1	0.549	0.857	0.819
	SVM	0.625	1	1	0.447	0.769	0.728
21em**5**	SRVM-Gauss	0.857	0.944	0.857	0.541	0.857	0.801
	SVM	0.857	0.888	0.75	0.519	0.8	0.718

Table 3.4: Alternative metrics for assessing the performance of the SRVM and SVM algorithms. These metrics take into account class imbalance in the training set, and are therefore a more robust and powerful measure of algorithm performance. The table makes clear that despite the statistically insignificant difference in accuracy between the algorithms, the SRVM method is consistently better across all metrics when class imbalance is accounted for.

somewhat similar to reinforcement learning [42], is to compute the overlap and/or accuracies for a set of parameters and then *recursively* use SRVM to extrapolate and decide on the optimal parameters. Since the accuracies/overlaps are continuous variables, this task lies in the domain of "regression" (the prediction of an outcome that is a continuous variable). We next briefly discuss regression more generally.

3.5 SRVR: Regression by Replication

Our natural functions f (and kernels) are continuous functions and thus, on an intuitive level, are more suggestively related to continuous value prediction (a regression) and not a discrete prediction (classification problems such as those that we

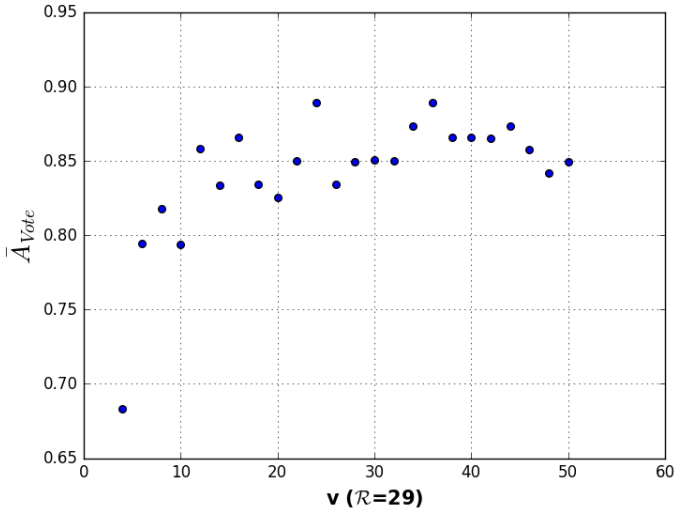


Figure 3.33: LSVT data set analyzed with layered voting (see Section 3.4.6) with a uniform weight, see Eq. (3.12). Accuracy as a function of the number of anchor points v for the Gaussian kernel. The number of replicas is held fixed at $\mathcal{R} = 29$. The accuracy is slightly higher than that of the single kernel (Fig. (3.9). The accuracy is expected to increase when the weights of the different voting kernels are optimized (and not set to a uniform equal values) as they are here).

considered in earlier Sections). As we will explain below, what is important for a regression solver is to have normal statistical properties. There are no "benchmarks" that are as clearly defined for continuous regression data as they are for discrete classifiers (where an answer is clearly wrong or right). With this in mind, we examined the features of the LSVT data set (for which we developed a binary classifier) and tested to see whether our predictors $f^{\alpha}(\vec{x})$ (sans the thresholding of Eq. (3.5) comprise good (continuous) regression predictors to the binary data.

The natural regression route for SRVM is to expand in kernels of the full vectors $\vec{x}$ (as in the kernels of Eq. (3.2) employed in the expansion of Eq. (3.1) that we have used thus far). However, the manner in which a regression is usually performed for other existing machine learning approaches is different. Instead of expanding in functions of all d components (features of a vector) $\vec{x}$, one expand in functions of individual features, e.g., for d features, one typically posits a function $\sum_{l=1}^{d} c_l f_l(x_l)$ (instead of considering functions of whole instance feature vectors $\vec{x}$); the reason that this single feature expansion is typically invoked is that multicollinearity is assumed

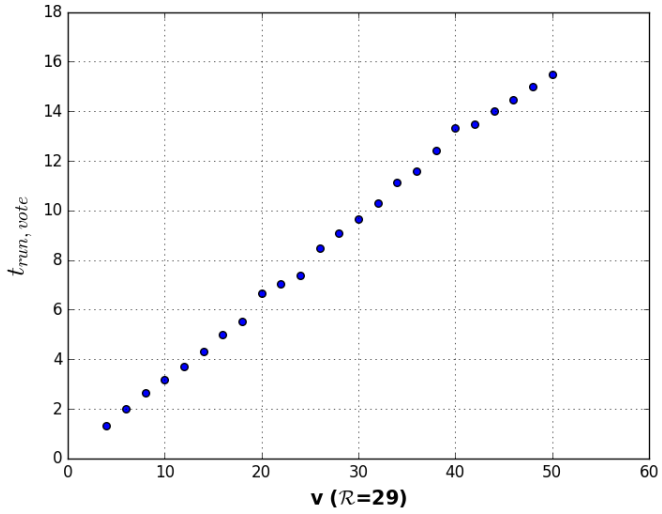


Figure 3.34: LSVT data set analyzed with layered voting; see caption of Fig. (3.33). Run time as a function of the number of anchor points for the Gaussian kernel.

and then this assumption may be consistently tested for. This also allows to test for the individual significance of a given feature. Rather explicitly, such a common regression amounts to an expansion of the form

$$y^{\alpha}_{i,p} \equiv f^{\alpha}(\vec{x}_i) \equiv \sum_{l=1}^{d} \sum_{j=1}^{v} c^{\alpha}_{lj}\ K_l(x_{li}, \{\vec{\chi}^{\alpha}_j\}), \tag{3.14}$$

where x_{li} denotes the l-th feature (component) of the vector $\vec{x}_i$.

In our work, we performed regression in both ways (expanding in kernels of the full d components vectors $\vec{x}$ and expanding in kernels of individual functions of the single components (single features) x_l). We mainly focused on the first one (that of expanding in functions of the full feature vector $\vec{x}$ as in Eqs (3.1, 3.2)). When searching for optimal parameters, one has to focus on the mean squared error versus the generalization error since the mean squared error (as seen in Fig. (3.36)) will always decrease with more anchor points v. However, the generalization error will increase dramatically when overfitting occurs beyond a threshold number of anchor points. Inspecting Fig. (3.36), we may ascertain the optimal number v of anchor points.

Testing a regression is notably more challenging than assessing the accuracy in a

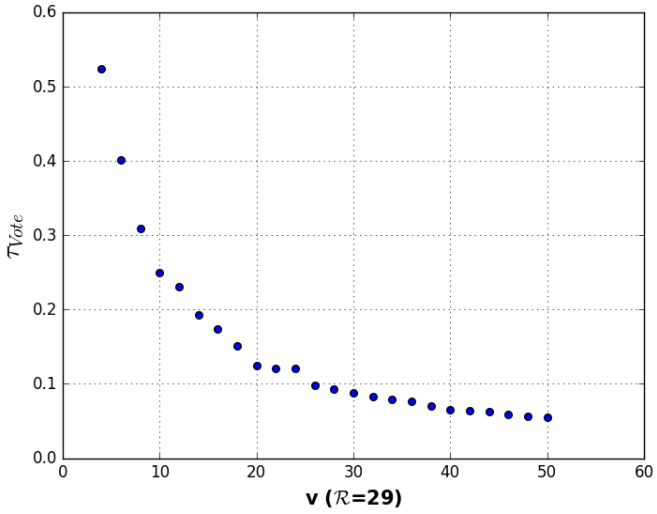


Figure 3.35: LSVT data set analyzed with layered voting; see caption of Fig. (3.33). Coefficient of performance (COP) as a function of the number of anchor points for the Gaussian kernel.

classification problem. The predicted outcome is clear cut for the discrete variable in a classification problem; this is obviously not the case for continuous functions. Instead of seeing whether the "exact" outcome is achieved (an impossible feat for continuous real numbers), additional, more detailed checks, are necessary. The commonplace minimization of the sum of square errors is indeed how we find the optimal parameters (that are used in the plots). The raw sum of square errors is not a useful metric on its own; it is more of a comparison metric (since it depends on the units and the number of data points).

In particular, for a regression to perform optimally, aside from predicting results that are close to the correct answers, its residuals (errors) should be random and normally distributed about the true population; the residuals should have no auto-correlation (no bias of one data point influencing another). In Figs. (3.37,3.38), we provide scatter plots and histograms of the residuals with the mean in red, standard deviations in blue and green. it is seen that the histograms are very normal. One may also look at probability plots (Fig. 3.39); apart from skewing at the tails, one sees good normal residuals. Autocorrelation statistical tests of the w value suggesting that no significant autocorrelations in the results- no bias in the regression. Multiple

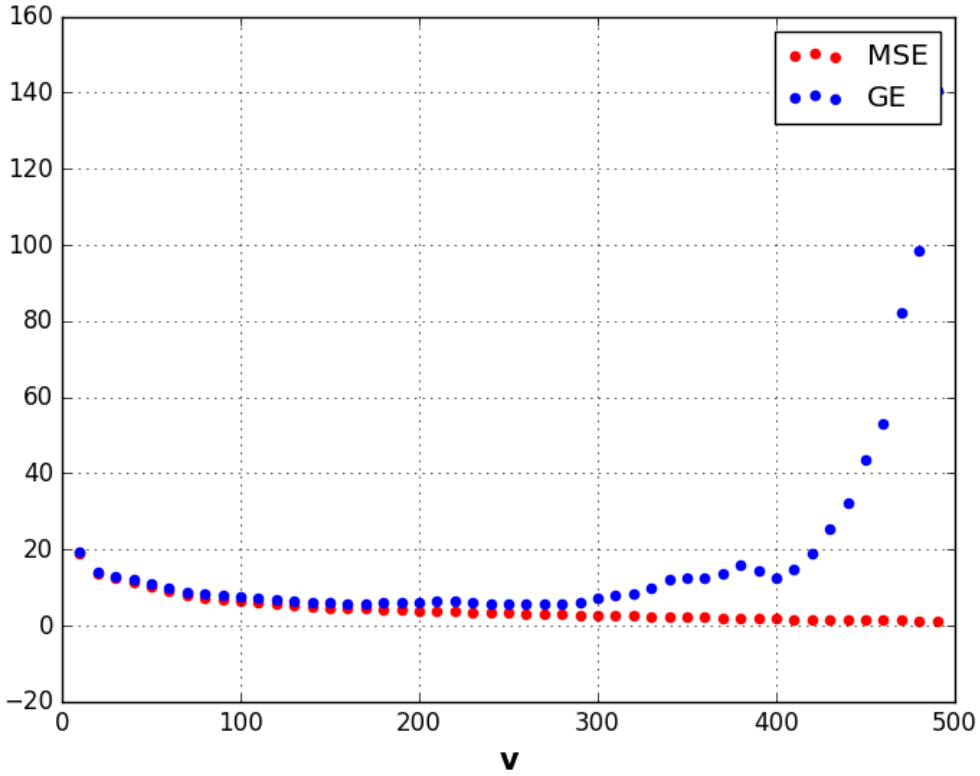
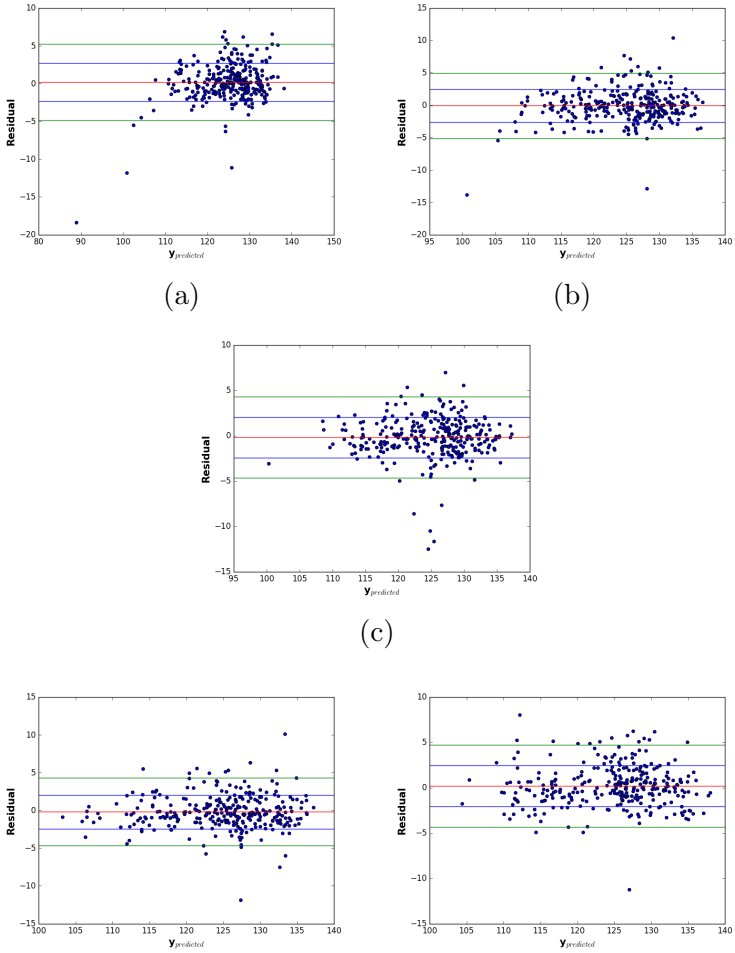


Figure 3.36: LVST data set. Plot of the mean standard training error (MSE) and the testing error (GE) for increasing number of parameters in the standard SRVMR algorithm. The optimal number of parameters corresponds to the value of v where GE and MSE have the lowest values for a single v. In this case that occurred at v=250.

layers of voting, the layers do not add up in the same way that they do for ordinary linear regressions. Indeed, one may not employ R^2 as a fair metric. There will a χ^2 statistic on the sum squares of errors (and relate to an F ratio).

3.6 Conclusion

In conclusion, we presented a new machine learning algorithm that we call "Stochastic Replica Machine Voting". The central premise of this approach is that the known data is fitted to fix the coefficients in multiple stochastic functions. Once these coefficients are fixed, predictions are made by the ensemble of these random functions

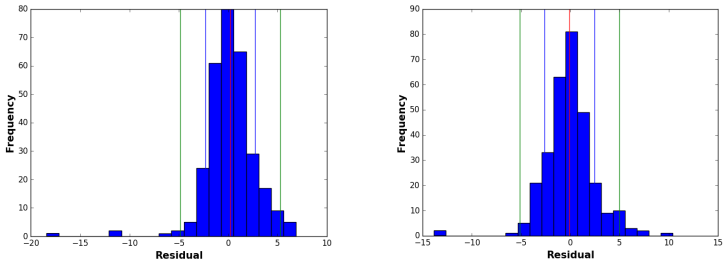


(a) (b)

(c)

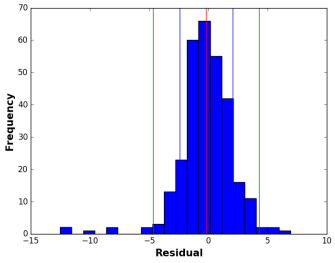
(d) (e)

Figure 3.37: (Color Online.) Scatter plots of the residuals vs predicted value for cross validation of the SRVM regression algorithm applied to the LSVT dataset with v=250 anchor points. The lines mark the locations of the mean (red), σ (blue) and 2σ (green). It is clear from the scatter plots, that the residuals of the SRVM regression model are randomly distributed and fall within the appropriate values (colored lines) for the normal distribution, suggesting accurate model performance.

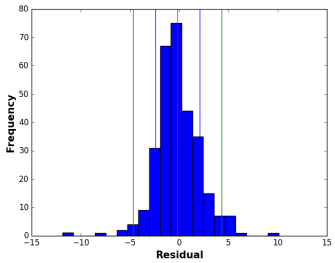


(a)

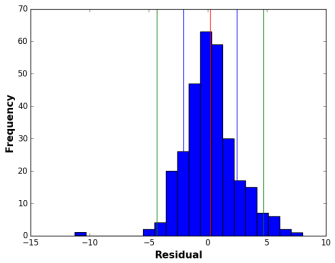
(b)



(c)



(d)



(e)

Figure 3.38: (Color Online.) Histograms of the residuals for five-fold cross validation testing of the SRVM regression algorithm for the LSVT dataset with v=250 anchor points. The vertical lines mark the locations of the mean (red), σ (blue) and 2σ (green). Based on the histograms, the residuals appear to be roughly normally distributed with the exception of some slight skewing in the tails. This result indicates a strong performance of the model.

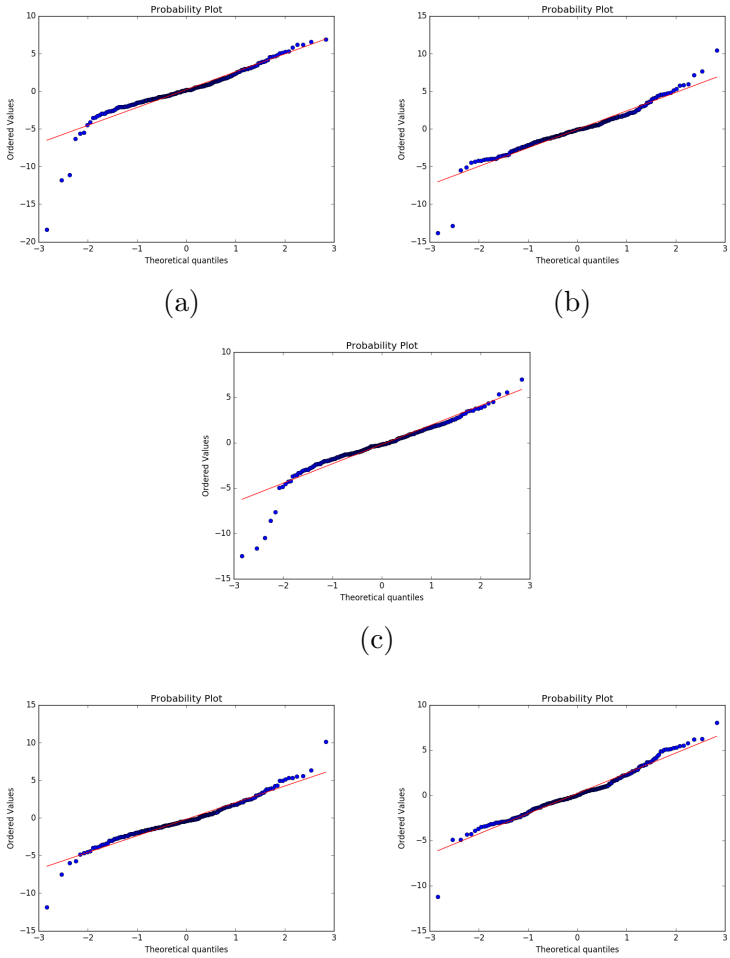


Figure 3.39: (Color Online.) Probability plots of the residuals for cross validation testing of the SRVM regression algorithm applied to the LSVT dataset with v=250 anchor points. The quantiles of the residuals are plotted against the expected quantiles of the normal distribution. With the exception of minor skewing in the tails, the quantiles appear normal, suggesting strong model performance.

as to the correct classification/regression of new data. Each of the functions "votes" for the predicted outcome. The system then averages over all predictions by weighing these in a chosen manner. We tested the algorithm's performance on multiple known benchmark problems. Overall, we found the accuracy of our algorithm to be comparable to that of standard well used techniques such as SVM. However, the optimal parameters in our model are set by using all of the given data (not tossing away a subset of these and using cross-validation). Rather, the optimal parameters are found by seeing when the different stochastic functions (the "replicas") have a high overlap- a consistency in their prediction regarding the outcome for particular data. No less notably, due to the intrinsic stochastic character of multiple functions used, the system is far superior to SVM in avoiding class imbalance bias. Given the simplicity of our algorithm and its natural extensions, much more work can be done to further streamline the algorithm and apply it to multiple data sets. Aside from the numerous sets tested here, two additional materials oriented (binary and ternary) classification problems were examined in [43].

Bibliography

[1] L. Einav and J. Levin, "Economics in the age of big data", Science Mag., **346**, 6210 (2014)

[2] https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

[3] A. Yildermaz, "Using Big Data to Decode Private Sector Wage Growth", arXiv:1609.09067 (2016)

[4] C. L. Philip Chen and C. -Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data", Information Sciences **275** (2014) 314?347

[5] A. Szalay and J. Gray, "Science in an exponential world", Nature **440** (2006) 23?24

[6] M. Hilbert and P. Lopez, "The World's Technological Capacity to Store, Communicate, and Compute Information", Science, **332**, Issue 6025, pp. 60-65 (2011)

[7] E. Alpaydin, *Introduction to Machine Learning, 3rd Ed.*, The MIT Press, ISBN: 978-0-262-02818-9 (2014).

[8] S. Fortunato, "Community detection in graphs", Physics Reports **486**, 75-174 (2010).

[9] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", Phys. Rev. E **69**, 026113 (2004).

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks", J. Stat. Mech. **10**, 10008 (2008).

[11] Peter Ronhovde and Zohar Nussinov, "An Improved Potts Model Applied to Community Detection", Physical Review E **81**, 046114 (2010).

[12] P. Ronhovde and Z. Nussinov, "Multiresolution community detection for megascale networks by information-based replica correlations", Phys. Rev. E **80**, 016109 (2009).

[13] . V. Gudkov, V. Montelaegre, S. Nussinov, and Z. Nussinov, "Community detection in complex networks by dynamical simplex evolution", Phys. Rev. E **78**, 016113 (2008).

[14] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure", Proc. Natl. Aca. Sci. U.S.A. **105**, 1118-1123 (2008).

[15] V. Vapnik, *The nature of statistical learning Theory, 2nd Ed* Springer, NewYork (1999).

[16] J.A.K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", Neural Processing Letters **9**, 293-300 (1999).

[17] D. J. Livingstone, *Artificial Neural Networks: Methods and Applications*, Humana Press (2011).

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, (2016), MIT Press, Cambridge, MA and London, England

[19] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", PNAS **79**;2554-2558 (1982)

[20] D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Spin-glass models of neural networks", Phys. Rev. A **32**, 1007 (1985)

[21] D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks", Phys. Rev. Lett. **55**, 1530 (1985)

[22] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model", Science **233**, 625 (1986).

[23] H. Sompolinky, "Statistical Mechanics of Neural Networks", Physics Today **41** (12), 70 (1988).

[24] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines", Cognitive Science **9**, 147-169 (1985)

[25] A. Karpatne, G. Atluri, J. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, "Theory-guided Data Science: A new paradigm for scientific discovery", arXiv: 1612.08544 (2016)

[26] K. Huang, *Statistical Mechanics*, John Wiley & Sons (New York, Chichester, Brisbane, Toronto, Singapore) (1987).

[27] L. D. Landau, "ON THE THEORY OF PHASE TRANSITIONS", Zh. Eksp. Teor. Fiz. 7, pp. 19-32 (1937).

[28] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation", Journal of Machine Learning Technologies. **2** (1): 37-63 (2011).

[29] A. Tsanas, M.A. Little, C. Fox, and L.O. Ramig, "Objective automatic assessment of rehabilitative speech treatment in Parkinson's disease", IEEE Transactions on Neural Systems and Rehabilitation Engineering, **22**, pp. 181-190, (2014), Data from: http://archive.ics.uci.edu/ml/datasets/LSVT+Voice+Rehabilitation

[30] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

[31] Statlog (Heart Disease) dataset, Lichman, M. (2013). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Statlog+(Heart)]. Irvine, CA: University of California, School of Information and Computer Science.

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

[33] K. Lakshminarayan, S. A. Harp, R. Goldman, and T. Samad, "Imputation of missing data using machine learning techniques", KDD-96 Proceedings (1996).

[34] Jose M. Jereza, Ignacio Molinab, Pedro J. Garcia-Laencinac, Emilio Albad, Nuria Ribellesd, Miguel Martin, and Leonardo Franco, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem", Artificial Intelligence in Medicine **50**, Issue 2, Pages 105-115 (2010).

[35] W. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis", Journal of the American Statistical Association. **47**, 583-621 (1952).

[36] Student (pseudo name of William Sealy Gosset), "The Probable Error of a Mean", Biometrika **6** 1:1-25 (1908)

[37] H. Levene, "Robust tests for equality of variances". In I. Olkin, S. S. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, "Contributions to Probability and Statistics: Essays in Honor of Harold Hotellin", Stanford University Press. pp. 278-292 (1960).

[38] https://en.wikipedia.org/wiki/Accuracy$_p aradox$

[39] J. Cohen, "A coefficient of agreement for nominal scales". Educational and Psychological Measurement **20** (1), 37-46.(1960); N. C. Smeeton, "Early History of the Kappa Statistic", Biometrics. **41**, 795 (1985).

[40] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation", Journal of Machine Learning Technologies **2** (1): 37-63 (2011).

[41] B. W. Matthews,"Comparison of the predicted and observed secondary structure of T4 phage lysozyme". Biochimica et Biophysica Acta (BBA) - Protein Structure **405** (2): 442-451 (1975).

[42] Richard Sutton and Andrew Barto,"Reinforcement Learning: An Introduction", MIT Press (1998).

[43] Bo Sun, T. Mazaheri, J. Scher-Zagier, D. Magee, P. Ronhovde, T. Lookman, and Z. Nussinov, https://arxiv.org/pdf/1705.08491.pdf

Chapter 4

Stochastic Replica Voting Machine Prediction of Stable Perovskite and Binary Alloys

4.1 Introduction

In recent years machine learning plays an more and more important role in the searches of new materials. Machine learning can make such searches far more efficient by systematically pointing to promising materials that may then be fabricated and tested experimentally. In this chapter, we will focus on the application of our Stochastic Replica Voting Machine on two kinds of material: Perovskite and binary alloys.

The first "Perovskite" ($CaTiO_3$) was named after the Russian mineralogist Lev Perovski. This name is, by now, used to describe an entire class of compounds of similar stoichiometry and structure [13] (see Fig. 1.5). Numerous, highly technologically important, materials are known display this structure [14]. High temperature superconductors exhibit Perovskite structure. Amongst their other applications,

Perovskites are employed as buffer/substrates that are heavily used in epitaxy, high-efficiency commercial photovoltaic [15], light-emitting diodes, in lasers, and many other systems, e.g.,[16, 17].

We introduce and summarize our new algorithm and demonstrate its application to the classification (viable formability) of (1) perovskite type compounds and (2) the classification of binary compounds. In both cases, we achieve high accuracy. Our results enable the prediction of new stable perovskites and the properties of binary compounds. Other works, e.g., [22, 23, 24] study various aspects of perovskites with existing machine learning algorithms. In the current work, we employed a new and very general machine learning algorithm (whose details will be reported on in [21]) and delineated new phase boundaries in the two classification problems that we investigated.

Our bare binary classifier can be trivially extended to non binary (multi-class) problems via, e.g., the "One-Versus-Rest" approach [25]. We will detail a 3-class problem when investigating two atom ("AB") alloys.

4.2 The Stochastic Replica Voting Machine algorithm

As befits its name, our "Stochastic Replica Voting Machine" (SRVM) algorithm relies on a voting procedure among stochastically generated classifiers. As we will explain, these individual classifiers are defined by a kernel that may be of any type: e.g., a sum of Gaussians or a multinomial. Initially, we "train" the system to predict the correct answer. The trained system may then subsequently predict the outcome given initial inputs. Training is performed by adjusting the kernel of each individual classifier such that it reproduces known results. The ensemble of classifiers is then given new data and a vote is taken amongst the predictions of the individual classifiers.

The input ("training set") data for N items that need to be classified is given in terms of a set of a vectors $\{\vec{x}_i\}_{i=1}^{N}$ defining the features of the items and their

corresponding classification σ_i. If the classification is amongst q different groups, then classification function is a Potts spin variable whose value $\sigma_i = 1, 2, \cdots, q$ denotes the group that item i correctly belongs to. Potts variables may be generally used as a classification index in numerous arenas, e.g., [27, 28, 29, 30]. The features of each item are combined into a vector $\vec{x} = (x_1, x_2, \cdots, x_d)$. Thus, the Cartesian components of each vector $\vec{x}_i$ are equal to the values of all parameters of the input data associated with item i (e.g., the values of the individual atomic radii of the ions forming in a candidate Perovskite material). If numerous features are given for each data point i, then the dimensionality (d) of the vectors $\vec{x}_i$ will be high. The goal of Machine Learning is to make an educated guess (a "prediction") as to what the corresponding classification outcome will be for a new vector $\vec{x}$ for which there is no a priori correct classification known outcome. Since no additional information is available, the predicted outcome σ can only be some function F of all supplied input: the features defining $\vec{x}$ and all known training set data. That is, the underlying assumption of any Machine Learning approach is that

$$\sigma(\vec{x}) = F(\vec{x}; \{\vec{x}_i\}_{i=1}^N, \{\sigma_i\}_{i=1}^N). \tag{4.1}$$

The natural question is "how may we determine the correct or 'optimal' function F"? Numerous Machine Learning approaches exist. We briefly comment on two of these. In one important subclass of these, known as "Support Vector Machines" (SVM), e.g., [31, 32], F is implicitly ascertained by inequalities applied to an assumed specific function types. In neural network based Machine Learning, in particular in "deep learning" [33], the function F is formed by a particular hierarchal recursive structure. Our approach (SRVM) is, in some regards, far more rudimentary than these and other prevalent models. To illustrate its basic premise, we will consider the binary (i.e., $q = 2$) classification problem. Here, $\sigma_i = 1, 2$ and thus $\tilde{\sigma}_i \equiv (2\sigma_i - 3) = \pm 1$ naturally classifies any data point $\vec{x}$ into one of two groups (labelled by $\tilde{\sigma}_i = 1$ and $\tilde{\sigma}_i = -1$). We define $\tilde{F} \equiv (2F - 3)$ and initially consider $\tilde{F}$ to be an outcome of a vote amongst the predictions of a large *ensemble* of stochastic functions $\{G_a\}_{a=1}^r$

where we term r to be the number of "replicas" in this ensemble. A simple choice for the functions G_a (that will be investigated in the current work) is one in which they are a sum of R random Gaussian functions [34]. Thus, we set

$$G_a = \sum_{j=1}^{R} c_{ja} e^{-(\vec{x}-\vec{x}_{ja})^2/(2\sigma_{ja}^2)}, \tag{4.2}$$

where $\{c_{ja}\}_{j=1}^{R}$ are coefficients that we will discuss momentarily. In the most minimal form of G_a, all standard deviations σ_{ja} are set to a uniform fixed value, $\sigma_{ja} = \sigma$. The centers $\{\vec{x}_{ja}\}$ of the Gaussians are randomly chosen in the volume spanned by the feature space. Thus, for each of the r functions $\{G_a\}_{a=1}^{r}$, we randomly choose R "anchor points" in the feature space volume to be $\{\vec{x}_{ja}\}_{j=1}^{R}$. The location of these anchor points differs from replica to replica. That is, we define each replica "a" by a different stochastic set of vectors $\{\vec{x}_{ja}\}_{j=1}^{R}$. More comprehensive than the specific choice of random Gaussians in Eq. (4.2), the functions G_a may be generally chosen to be of the form

$$G_a = \sum_{j=1}^{R} c_{ja} K_a^j(\vec{x}). \tag{4.3}$$

Here, the kernel (or basis) functions K_a^j may be an arbitrary stochastic functions. For the Gaussian form of Eq. (4.2), $K_a^j(\vec{x}) = e^{-(\vec{x}-\vec{x}_{ja})^2/(2\sigma_{ja}^2)}$. Other general kernels K, different from a Gaussian function, may, of course, be considered. For instance, another natural (yet typically computationally expensive) choice for the kernel K_a that we will return to in the current work (reasonable when the outcome likelihood is expected to be analytic as a function of the features) is that of multinomials in the Cartesian components of $\vec{x}$.

During the training phase, we optimize the values of the coefficients $\{c_{ja}\}_{j=1}^{R}$ given the known outcome for the training points $i = 1, 2, \cdots, N$. The number R of the coefficients required in order to achieve high prediction accuracy, is typically smaller than the number of training data points, $R < N$ (in most instances, in fact, $R \ll N$). The optimal value of R depends on the nature of data as well as the size of data

and should be chosen carefully to avoid over-fitting. For each replica, $a = 1, 2, \cdots, r$, the given training data set translates into linear equations for $\{c_{ja}\}_{j=1}^{R}$. Thus, Eq. (4.3) explicitly reads $G_a(\vec{x} = \vec{x}_i) = \sum_{j=1}^{R} K_a^{ij} c_{ja}$, where $K_a^{ij} \equiv K_a^j(\vec{x} = \vec{x}_i)$. This embodies a set of overdetermined (since $N > R$) linear equations for the coefficients $\{c_{ja}\}$. For each of the replicas $a = 1, 2, \cdots, r$, the above relation can be trivially cast as an explicit matrix equation, $\hat{G}_a = \hat{K}_a \hat{c}_a$. Here, $\hat{G}_a$ and $\hat{c}_a$ are two column vectors of, respectively, lengths N and R whose entries are, respectively, $\{G_a(\vec{x} = \vec{x}_i)\}_{i=1}^{N}$ and $\{c_{ja}\}_{j=1}^{R}$. The elements of the rectangular $N \times R$ dimensional matrix K_a are, as defined above, given by $(\hat{K}_a)^{ij} \equiv K_a^j(\vec{x} = \vec{x}_i)$. The coefficients $\hat{c}_a$ minimizing the square sum $||\hat{G}_a - \hat{K}_a \hat{c}_a||^2$ are given by

$$c_{ja} = \sum_i (\hat{K}_a^{-1})_{ji} G_{ia}. \tag{4.4}$$

Here, the rectangular matrix $\hat{K}_a^{-1}$ (with elements $(\hat{K}_a^{-1})_{ji}$) is the pseudoinverse of $\hat{K}_a$. Thus, in the training phase, the goal is to find the coefficient vectors $\hat{c}_a$ for each of the replicas $a = 1, 2, \cdots, r$. With the above values of c_{ja} in tow, we may now predict the classification of a new "test" item $\vec{x}$ different from all prior training data points (i.e., $\vec{x} \neq \vec{x}_i$ for $1 \leq i \leq N$). That is, we may compute the classification of $\vec{x}$ as predicted by the r independent replicated stochastic functions, $\{sgn(G_a(\vec{x}))\}_{a=1}^{r}$ (where sgn denotes the sign function) and then perform a vote amongst all of these classifiers. The vote then yields the final prediction of the SVRM,

$$\tilde{\sigma}(\vec{x}) = sgn(\sum_{a=1}^{r} sgn(G_a(\vec{x}))). \tag{4.5}$$

For the $q = 2$ classification problem that we considered thus far, the inner sgn functions in Eq. (4.5) may be replaced by other appropriately chosen symmetric functions W, i.e., $\tilde{\sigma}(\vec{x}) = sgn(W(G_1, G_2, \cdots, G_r))$; the single condition on $\tilde{\sigma}$ is that its value may only be either 1 or (-1) (corresponding to the two possible classes to which an item $\vec{x}$ may belong to).

Putting all of the pieces together, Eqs. (4.3,4.4,4.5) nearly completely define the

SRVM program. The kernels K_a^j may, a priori, stochastically be chosen to be of any particular functional form. Of course, if an existing theory exists then the functional form of G_a may be more efficiently designed. In the absence of any such information, one may simply examine the predictions for random kernels K_a^j. There are three remaining inter-related natural questions:

(1) Is there a particular metric to determine the confidence with which the results are predicted?

(2) How do we determine the 'optimal' number r of the replicas to be used?

(3) Similarly, what sets the number R of kernel functions in Eq. (4.3)?

As we will describe, the answer to all questions may be determined by examining the overlap of the predictions of the different stochastic replica functions $\{G_a\}_{a=1}^r$. Throughout the current work, we will employ a simple variant of the overlap $\mathcal{O}(\vec{x})$ associated with any point $\vec{x}$ whose classification is predicted by the r replicas $\{G_1, G_2, \cdots, G_r\}$, namely

$$\mathcal{O}(\vec{x}) \equiv \frac{1}{r} |\sum_{a=1}^{r} sgn(G_a(\vec{x}))|. \tag{4.6}$$

With this definition in tow, we first explicitly turn to question **(1)**. If all replicas yield identical predictions (and thus $\mathcal{O}$ is close to unity), then (as is intuitively natural and we verified by numerical experiments), this common predicted answer is likely correct. Analogously, if the replicas are far from a unanimous agreement about the correct classification (and, consequently, $\mathcal{O}$ is much smaller than one) then the predicted answer cannot be trusted with high confidence. The above rule of thumb enables us to scan the parameters r and R to find values that are likely to yield optimal accuracy (questions **(2)** and **(3)** above). Typically as the number of replicas r increases so does the accuracy. However, larger values of r entail increasing computational costs

with no real benefit. We thus seek sufficiently large r that enable high accuracy. By contrast, when the number of anchor points (or more general basis functions) R becomes too large, overfitting leads to increasing errors. There are optimal values of R that are sufficiently large to capture the characteristics of the data yet not so big that overfitting occurs. In reality, we may fix r and R to specific values and examine the replica overlap to ascertain whether the predicted values may be trusted [21]. That is, when the overlap $\mathcal{O}$ is averaged over all new data points $\vec{x}$ (whose correct classification is not a priori known and that need to be classified by the algorithm) is high, then the consensus reflected by the average $\mathcal{O}$ will suggest that the current parameters r and R defining Eqs. (4.3, 4.6) enable a correct prediction of the classification problem.

A variant that we will touch on later is that of "an expansion in a box". For typical basis functions K_a^j, the functional form of Eq. (4.3) assumes that the outcome is a generally smooth function of $\vec{x}$. If the system exhibits "phase transitions" as a functions of the features $(x_1, x_2, \cdots, x_d)$ then such an assumption is void. Instead, one may fit the training data with a particular function of the form of Eq. (4.3) with specific coefficients $\{c_{ja}\}$ only when $\vec{x}$ lies in a particular volume, $\vec{x} \in \Omega$; different regions will correspond to different functional forms (i.e., the coefficients $\{c_{ja}\}$ may change from one region of $\vec{x}$-space to another). Here, the expansion will be valid only in a particular "box". The function G_a will be allowed to change as $\vec{x}$ goes from being in one domain Ω to another. Thus, in each of the domains $\{\Omega_b\}$ comprising the system (in which the system is assumed to be "analytic") there will be a different function G_{ab} (specified by coefficients c_{jab}). In these cases, a natural question is how to ascertain phase transitions and effectively employ the existence of these volumes. Our approach here is once again that of noting when the overlap between different replicas is highest. That is, given a particular test point $\vec{x}$, we may train the system with all data that lies in a volume Ω (a "box") that encloses $\vec{x}$. We then see when, as a function of the size $||\Omega||$, the overlap $\mathcal{O}(\vec{x})$ between the replicas for the predicted outcome at point $\vec{x}$ will be the highest. We employed this approach when the overlap between the various replicas was small and the our original classification outcome was less certain.

The accuracy of Machine Learning classification algorithms is typically tested by randomly fitting a fraction z of the known data (i.e., using these data for "*training*") and then seeing how well the algorithm correctly predicts the classification of the remaining data that are not used as training but rather supplied to the algorithm only as new vectors $\vec{x}$ whose correct classification is known yet not given to the user but is to be predicted by the algorithm. This process (or training with a fraction z of the data and *testing* the predictions on the remaining fraction of $(1-z)$) is repeated over and over again with different ways of splitting the known data into two subgroups of relative numbers set by a parameter z,

$$\begin{aligned} \text{training data points} &: \text{test data points} \\ = z &: (1-z). \end{aligned} \tag{4.7}$$

The accuracy of the predicted classification is then averaged over the many ways of splitting the data with this ratio between the size of the number of training data points and the tested points kept fixed. In the accuracy tests that we will report on, we will follow the prevalent practice of choosing $z = 0.8$.

4.2.1 Gaussian kernels

In what follows, we provide an explicit example in which the value of R (the number of basis functions) is determined. In the current context, we seek to find the optimal number R of anchor points for the Gaussian of Eq. (4.2). Towards this end, we may plot the average overlap $\mathcal{O}$ between different replicas as a function of the number of replicas r and the number of anchor points R. This overlap enables us to determine the optimal values of r and R for which $\mathcal{O}$ obtains its maximum (or, more generally, its maxima).

To illustrate the basic premise, we examine the data of the Perovskite classification problem that we will turn to in greater detail later on. For the time being, we probe how the average of the overlap $\mathcal{O}$ varies as a function of the number of basis functions used (or anchor points in the case of the Gaussian kernel of Eq. (4.2)). (As remarked

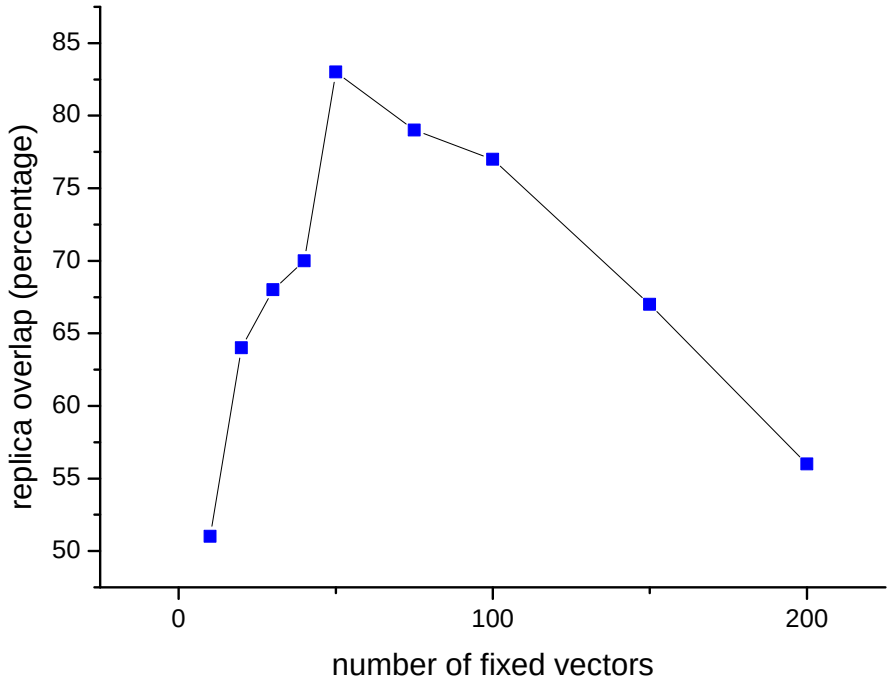


Figure 4.1: Overlap between different replicas (Eq. (4.6)) when the Gaussian kernel was in cubic Perovskite classification problem

earlier (see discussion after Eq. (4.2), the anchor points are randomly placed in the feature space.) As Fig. (4.1) illustrates, the overlap between different replicas is maximal for $R \approx 60$ anchor points. Since the inter-replica overlap is maximal for this value of R, we suspect using this number of anchor points would result in the optimal accuracy. The average accuracy that we reached with the Gaussian kernel for determining stable Perovskite oxides was 94.19 %. This accuracy may be contrasted with the performance of the current state of the art SVM package [35]; the SVM method yielded a mean accuracy of 92.53 %.

4.2.2 Multinomial kernels

As we alluded to earlier, another set of kernels in Eq. (4.3) is afforded by a $d-$component vector $\vec{j}$ defining monomials,

$$K^{\vec{j}}(\vec{x}) = x_1^{j_1} x_2^{j_2} ... x_d^{j_d}. \tag{4.8}$$

Here, x_k are the Cartesian components ($1 \leq k \leq d$). There are a variety of ways to produce multinomial based replica. For instance, different rotations in parameter space may lead to independent multinomials. A general rotation $x_k \to U^h_{kk'} x_{k'} \equiv x_{kh}$ with U^a a random rotation matrix, will transform the monomial of Eq. (4.10) into multinomial in which the sum of all powers in each of the individual monomials

$$J \equiv \sum_{k=1}^{d} j_k \tag{4.9}$$

is unchanged relative to its value in Eq. (4.10). Thus, if we choose a basis of monomials $\{K^{\vec{j}}_a(\vec{x})\}$ with $0 \leq j_k \leq p$ (with a general natural number p) for all $1 \leq k \leq d$ in one coordinate system x_k then an independent basis of monomials is afforded by

$$K^{\vec{j}}_a = x_{1a}^{j_1} x_{2a}^{j_2} ... x_{da}^{j_d}, \tag{4.10}$$

with $j_k \leq p$. This is so as the highest power of each of the Cartesian coordinates is $p < J$. In Eq. (4.10), $\{x_{hk}\}_{k=1}^{d}$ are the coordinates in the rotated basis generated by U^a. Eq. (4.3) may be used to concoct several replica functions $G_a = \sum_{j=1}^{R} c_{ja} K^{\vec{j}}_a(\vec{x})$.

4.2.3 Ternary and multi-class SRVM

Thus far, we focused on binary classification (wherein the sign (Eq. (4.5)) decided to which of two categories a particular point $\vec{x}$ should belong to. There is, of course, more to life than only binary classification. In order to classify $\vec{x}$ into one of $p > 2$ groups, various constructs are possible. One, very rudimentary, design is to iteratively classify as a point $\vec{x}$ as belonging (or not) to any one of the classes $q = 1, 2, \cdots, p$.

Such a rudimentary approach emulates the well known "One-Versus-all" (OVR) [25] technique; this is the what we will adopt in the current work when we will classify AB solids into one of $p = 3$ groups (Section 4.4). Specifically, we will start by predicting the results of an input vector $\vec{x}$ for each of the q possible output values with the SRVM algorithm that we introduced in the earlier subsections. Similar to the binary classification, in order construct the pseudo-inverse for the i-th output value bivariate algorithm, we will set the result of a data point as +1 if it output the i-th output value, and to be -1 otherwise. Instead of just taking the sign of the outputted results, we compared the raw values from results. That is, if the output associated with the vector $\vec{x}_i$ as tested against candidate classes $q = 1, 2, \cdots, p$ had the highest incidence of positive values for a particular class $q = q'$ then the vector $\vec{x}_i$ was classified as belonging to group q'.

4.3 Perovskite formability

In this section we employ SRVM to predict whether candidate ABX_3 compounds form stable Perovskite structures.

The training data that we used [19] has $d = 2$ features:
(i) The "tolerance factor",

$$x_1 \equiv \frac{r_A + r_X}{\sqrt{2}(r_B + r_X)} \tag{4.11}$$

where $r_{i=A,B,X}$ denote, respectively, the radii of the A, B, and X ions, and
(ii) The "octahedral factor" defined as the ratio

$$x_2 \equiv \frac{r_B}{r_X}. \tag{4.12}$$

The data in [19] consist of 223 candidate compounds of the ABX_3 type. Of these compositions, 34 correspond to stable Perovskite structures and the rest are unstable structures. (After removing duplicate compounds that share the same tolerance

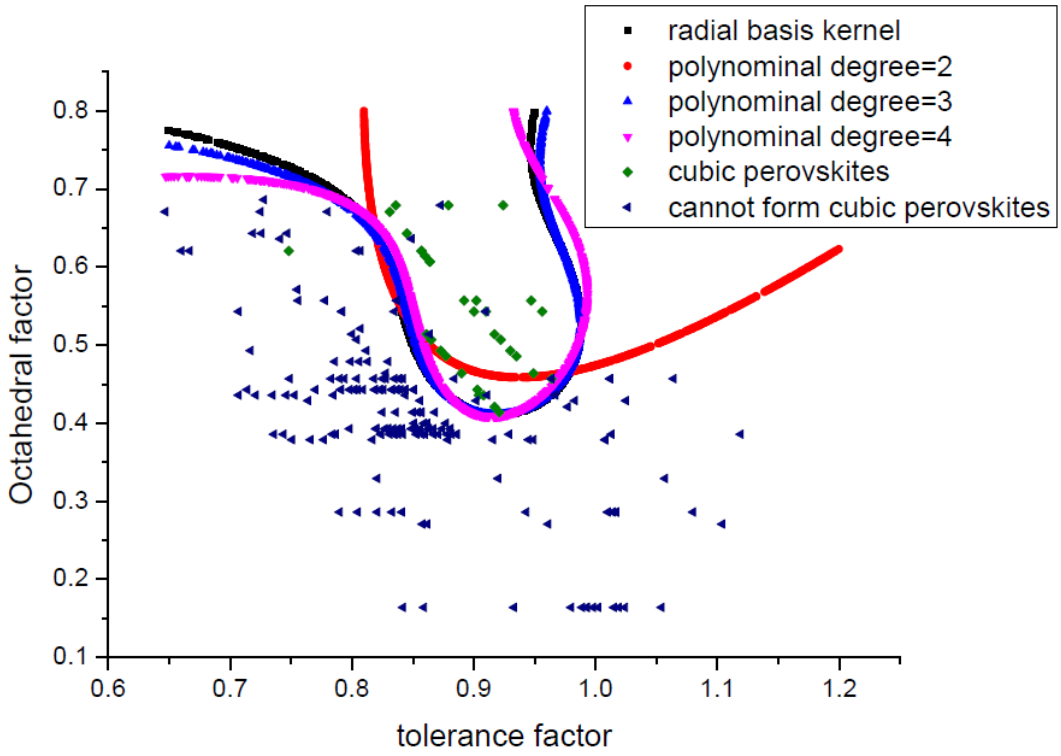


Figure 4.2: Classification results using different SVM kernels employing the Libsvm-3.0 package. [35]

factor and octahedral factor, 188 data points remain, 29 of which form stable cubic Perovskite structure.) Once the training is performed with input data, we use it to make the binary prediction regarding the stability of the contending Perovskite compounds. Following Eq. (4.7), we repeatedly partitioned the data into two random subgroups with $z = 0.8$. Several partitions with this ratio were generated by the standard cross-validation method in which the data is divided into nearly five equal parts. Subsequently, four of these five sets are then used together to train the algorithm and the remaining one fifth of the data is used as a resource of test data to see how accurate the predictions of the algorithm are. The set that is used as the test data is cycled through (being chosen to be all of the five nearly equal parts of the data). The accuracy is then averaged over the predictions made over the five groups when these are used as test data. The accuracy is further averaged over different random partitions into five groups. Both for comparison as well as in order to obtain a more comprehensive picture, aside from our own SRVM algorithm, we

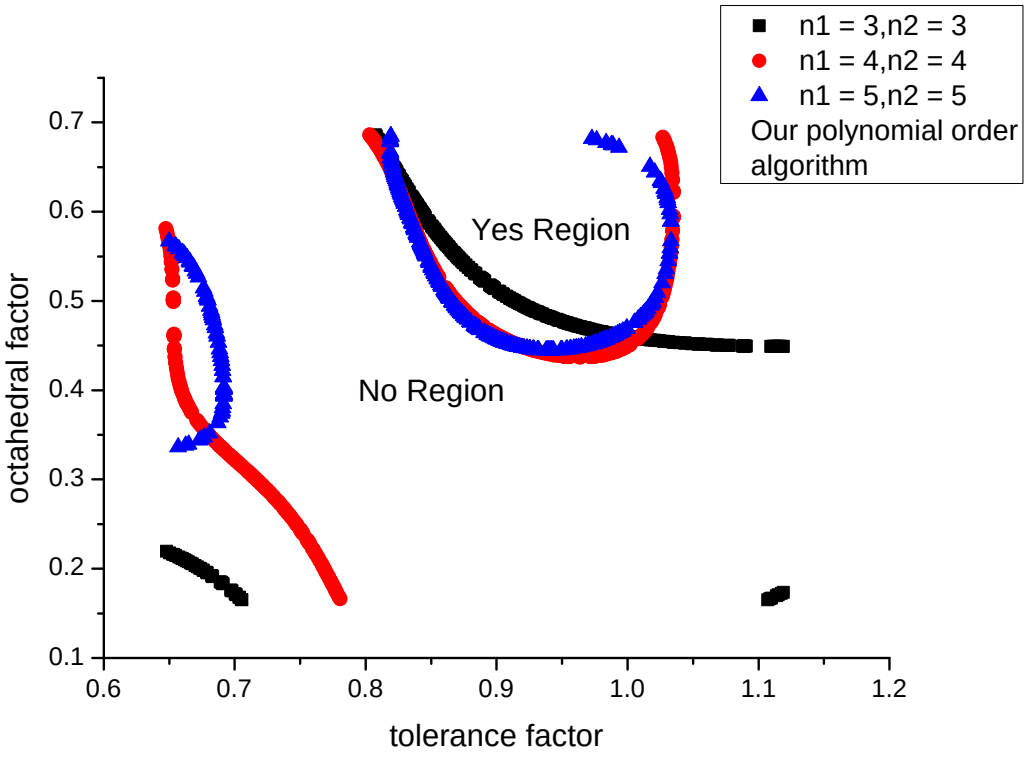


Figure 4.3: The viable region in the x_1x_2 plane for materials that may form cubic Perovskite structure as ascertained by a multinomial order kernel in the SRVM method . Here we employed multinomials of three different orders (3, 4, and 5). The region in which all multinomials predict formability of a cubic Perovskite structure is the common "Yes" region.

also employed both the standard Gaussian and polynomial kernels in the well known SVM method [31, 32]. In Figure 4.2, we provide the results that we obtained by applying the SVM algorithm for different kernels. In this figure, the region above the drawn curves (associated with individual SVM kernels) is predicted to correspond to stable Perovskite structures; in the parameter region below these curves, no stable Perovskite materials are anticipated. In line with our main dissertation (that of inferring a likely outcome from multiple independent kernels), the region that is above all drawn curves corresponds to a domain in the x_1x_2 plane in which we may expect (with high confidence) stable Perovskite structures. Similarly, in Figures 4.3, 4.4, we display the results obtained by our SRVM algorithm for, respectively, the multinomial and Gaussian kernels respectively (see Section 4.2 and the discussion following

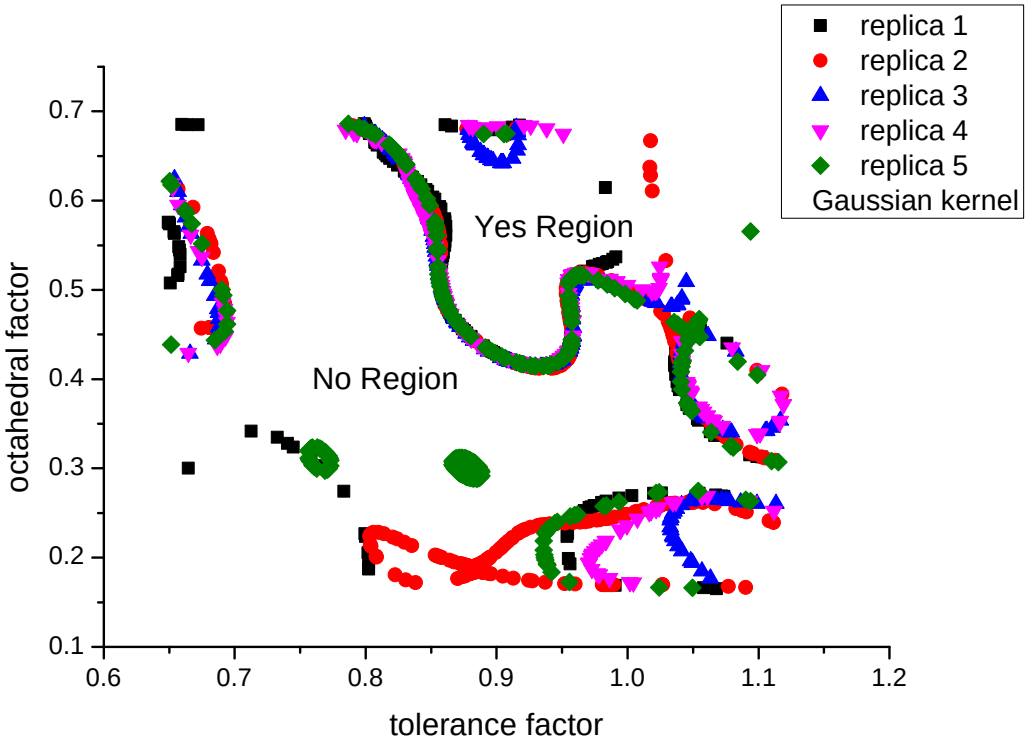


Figure 4.4: The predicted formability of the cubic Perovskite structure as provided by the Gaussian kernel. We find the common "Yes" region by using five different replicas. These different replicas are produced by randomly choosing 50 fixed vectors (see text).

Eq. 4.2 for a description of replicas in the Gaussian case). The designations of "Yes" and "No" reflect the predictions of the algorithm regarding the viability of putative compounds of an ABX_3 type composition to form stable Perovskite structures. In Figure 1.6, we overlay (with different levels of resolution in the two panels) the predictions of the SVM method and our SRVM algorithm with multiple kernels/replicas. The shaded region in Figure 1.6 is the one in which all methods/replicas/functions predict that stable Perovskite structure should form. With this region in hand, all candidate ABX_3 materials (of the correct chemistry to allow such a composition) with tolerance and octahedral factors that lie in the shaded area are predicted to be stable Perovskites. Some compositions lie near the boundary and do not enable (insofar as our approach is concerned) a definite prediction regarding new Perovskite structures. Two such candidates are $EuZrO_3$ ($x_1 = 0.857,\ x_2 = 0.514$) and $EuHfO_3$ ($x_1 = 0.861,\ x_2 = 0.507$). The location of these new potential stable Perovskite structures is highlighted in panel (b) of Figure 1.6. With $z = 0.8$, the SVM algorithm achieved an accuracy of 92.52%. By contrast, the SRVM algorithm obtained an accuracy of 94.14 % with a multinomial kernel (here two different multinomials (where different order multinomials were used as replicas) and we further employed the "expansion in the box" construction) and an accuracy of 94.19% with a Gaussian kernel (here we employed 11 replicas each having randomly chosen anchor points).

4.4 Ternary classifications of AB solids

We next turn our attention, using the data of [36], to the classification of binary solids [26] (of chemical composition AB) into one of $q = 3$ groups (denoted W, Z or R [26, 36]). Similar to Section 4.3, we applied both the standard SVM technique with the multi-class variant our SRVM approach (see Section 4.2.3) with multinomial and

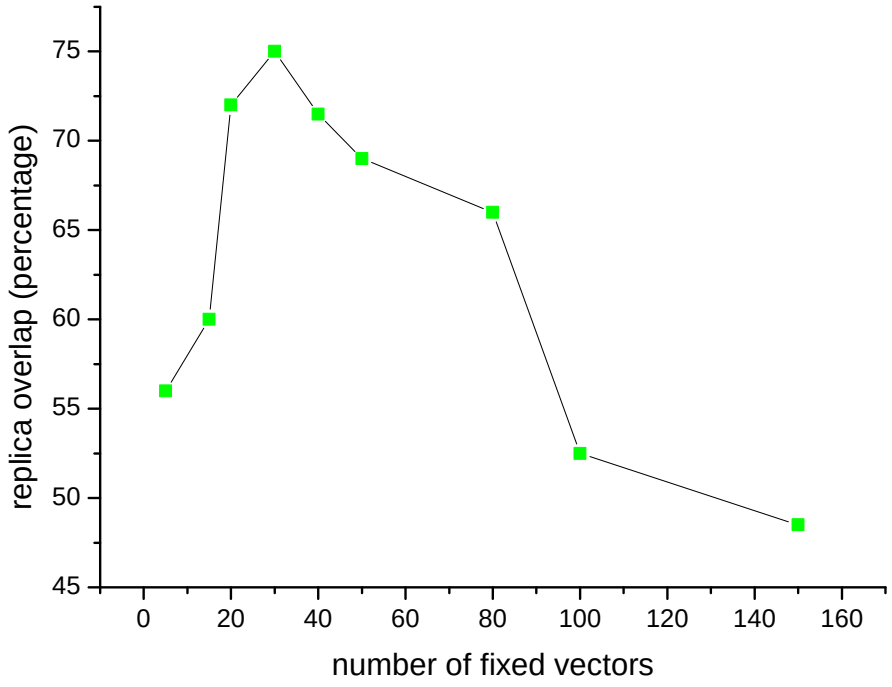


Figure 4.5: Overlap between different replicas (Eq. (4.6)) with Gaussian fit in the classification of the AB solids.

Gaussian kernels. We employed two commonly used figures of merit as features,

$$r_\sigma \equiv r_s^A + r_p^A - r_s^B - r_p^B,$$
$$r_\pi \equiv r_p^A - r_s^A + r_p^B - r_s^B. \tag{4.13}$$

Here, r_s^A, r_p^A, r_s^B and r_p^B denote the pertinent radii for an electron bound to the A or B ion that is in an s or p orbital.

In the Gaussian approach, we employed the sum of $R = 30$ individual Gaussians (associated with different anchor points). The general behavior of the inter-replica overlap is displayed in Figure 4.5 in which it is seen that the overlap becomes maximal at $R \sim 30$. The final classification for each data point was determined by the group for which a given data point appeared most frequently out of the five replicas employed. The cross-validation accuracies (as ascertained by Eq. (4.7) for $z = 0.8$) that our SRVM algorithm obtained for the Gaussian and multinomial kernels were,

respectively, 92.72% and 90.90%. These values were lower than the accuracy achieved by an SVM algorithm with a radial kernel (that we found to be 94.54%). In Figure 1.7, we provide the phase boundaries (between the W, Z, and R phases) as ascertained by SVM (see the solid curves therein) alongside the boundaries determined by our SRVM method (the domains of the different phases as predicted by SRVM are marked by different colors).

4.5 Conclusion

In this work, we introduced and implemented a new classification algorithm to classify various materials. In particular, we investigated (1) the formability of perovskite type compounds (a binary classification problem) and (2) classified AB type systems (via ternary classification). A more detailed description of our new algorithm appears in a companion paper [21]. Using this algorithm, we achieved a high accuracy in both problems and suggested new candidate stable perovskites and properties of binary compounds.

Bibliography

[1] T. Mueller, A. G. Kusne, and R. Ramprasad, *Machine Learning in Materials Science, in Reviews in Computational Chemistry*, volume **29**, Edited by A. L. Parrill and K. B. Lipkowitz, John Wiley & Sons, Inc, Hoboken, NJ. doi: 10.1002/9781119148739, chapter 4 (2016).

[2] *Information Science for Materials Discovery and Design*, Springer Series Materials, volume **225**, Edited by Turab Lookman, Frank Alexander and Krishna Rajan. 978-3-319-23870-8 (2016).

[3] P. Ronhovde, S. Chakrabarty, M. Sahu, K. F. Kelton, N. A. Mauro, K . K. Sahu, and Z. Nussinov, *Detecting hidden spatial and spatio-temporal structures in glasses and complex physical systems by multiresolution network clustering*, The European Physics Journal E **34**, 105 (2011).

[4] Prasanna V. Balachandran, Scott R. Broderick, and Krishna Rajan, *Identifying the 'inorganic gene' for high-temperature piezoelectric perovskites through statistical learning*, Proc. R. Soc., DOI: 10.1098/rspa.2010.0543 (2011).

[5] P. Ronhovde, S. Chakrabarty, M. Sahu, K. K. Sahu, K. F. Kelton, N. Mauro, and Z. Nussinov *Detection of hidden structures on all scales in amorphous materials and complex physical systems: basic notions and applications to networks, lattice systems, and glasses*, Scientific Reports **2**, 329 (2012).

[6] J. C. Snyder, M. Rupp, K. Hansen, K. R. Muller, and K. Burke, *Finding density functionals with machine learning*, Physical Review Letters **108** (25), 253002 (2012).

[7] Yi Zhang and Eun-Ah Kim, *Quantum Loop Topography for Machine Learning*, arXiv:1611.01518 (2016).

[8] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, Nature Physics **13**, 431 (2017).

[9] G. Carleo and M. Troyer, *Solving the quantum many-body problem with artificial neural networks*, Science **355**, 602 (2017).

[10] Maciej Koch-Janusz and Zohar Ringel, *Mutual Information, Neural Networks and the Renormalization Group*, arXiv:1704.06279 (2017).

[11] G. Pilania, A. Mannodi-Kanakkithodi, B. P. Uberuaga, R. Ramprasad, J. E. Gubernatis and T. Lookman, *Machine learning bandgaps of double perovskites*, Scientific Reports **6**, 19375 (2016).

[12] Arun Mannodi-Kanakkithodi, Ghanshyam Pilania, Tran Doan Huan, Turab Lookman, and Rampi Ramprasad, *Machine Learning Strategy for Accelerated Design of Polymer Dielectrics*, Scientific Reports **6**, 20952 (2016).

[13] O. Muller and R. Roy, *The Major Ternary Structural Families*, Springer, New York (1974).

[14] Alexandra Navrotsky, *Energetics and Crystal Chemical Systematics among Ilmenite, Lithium Niobate, and Perovskite Structures*, Chem. Mater. **10**, 2787?2793 (1998) DOI: 10.1021/cm9801901.

[15] N. -G. Park, *Perovskite solar cells: An emerging photovoltaic technology*, Materials Today **18**, 65-72 (2015).

[16] F. S. Galasso, *Perovskites and High Tc Superconductors*, Gordon and Breach, New York (1990).

[17] Z. L. Wang and Z. C. Kang, *Functional and Smart Materials: Structural Evolution and Structure Analysis* Plenum Press, New York (1998).

[18] http://reflexions.ulg.ac.be/cms/c$_3$98977/*en/nano − architects?portal = j$_5$5printView = true*

[19] L. M. Feng, L. Q. Jiang, M. Zhu, H.B. Liu, X. Zhou, and C .H. Li, *Formability of ABO3 cubic perovskites*, Journal of Physics and Chemistry of Solids **69**, 967 (2008).

[20] A. Khan and S. G. Javed, *Predicting regularities in lattice constants of GdFeO3-type perovskites*, Acta Cryst. Section B: Structural Science **64**, no. 1, pp. 120 (2008).

[21] Patrick Chao, Tahereh Mazaheri, Zohar Nussinov, Bo Sun, and Nicholas B. Weingartner, *A stochastic replica-based voting algorithm for supervised learning*, to appear.

[22] Syed Gibran Javeda, Asifullah Khanb, Abdul Majidb, Anwar M. Mirzac, and J. Bashir, *Lattice constant prediction of orthorhombic ABO3 perovskites using support vector machines* Computational Materials Science **39**, 627 (2007).

[23] Geoffroy Hautier, Christopher C. Fischer, Anubhav Jain, Tim Mueller, and Gerbrand Ceder, *Finding Nature's Missing Ternary Oxide Compounds Using Machine Learning and Density Functional Theory*, Chem. Mater. **22**, 3762 (2010), DOI:10.1021/cm100795d.

[24] G. Pilania, P. V. Balachandran, J. E. Gubernatis, and T. Lookman, *Classification of ABO3 perovskite solids: a machine learning study*, Acta Crystallographica Section B **71**, 507 (2015).

[25] Ryan Rifkin and Aldebaro Klautau, *In Defense of One-Vs-All Classification*, Journal of Machine Learning Research (JMLR) **5**, 101 (2004).

[26] Prasanna V. Balachandran, James Theiler, James M. Rondinelli and Turab Lookman, *Materials Prediction via Classification Learning*, Scientific Reports **5**, 13285 (2015).

[27] Shai Wiseman, Marcelo Blatt, and Eytan Domany, *Superparamagnetic clustering of data*, Phys. Rev. E **57**, 3767 (1998).

[28] Peter Ronhovde and Zohar Nussinov, *An Improved Potts Model Applied to Community Detection*, Physical Review E **81**, 046114 (2010).

[29] Peter Ronhovde and Zohar Nussinov, *Multiresolution community detection for megascale networks by information-based replica correlations*, Phys. Rev. E **80**, 016109 (2009).

[30] D. Hu, P. Ronhovde, and Z. Nussinov, *A Replica Inference Approach to Unsupervised MultiScale Image Segmentation*, Phys. Rev. E **85**, 016101 (2012).

[31] V. Vapnik, *The nature of statistical learning Theory*, 2nd edition, Springer, NewYork (1999).

[32] J.A.K. Suykens and J. Vandewalle, *Least squares support vector machine classifiers*, Neural Processing Letters **9**, 293 (1999).

[33] G. E. Hinton, S. Osindero, and Y. Teh, *A* fast learning algorithm for deep belief nets, Neural Computation **18**, 1527 (2006).

[34] A trivial extension of these functions is one in which the Cartesian coordinate axis are rotated and dilated (allowing for a sum of general multivariate Gaussian distributions with a non-diagonal kernel). For simplicity, in the current work, we will contend ourselves with the simple diagonal covariant form of Eq. (4.2).

[35] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

[36] Yousef Saad, Da Gao, Thanh Ngo, Scotty Bobbitt, James R. Chelikowsky, and Wanda Andreoni, *Data mining for materials: Computational experiments with AB compounds*, Phys. Rev. B **85**, 104104 (2012).