JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

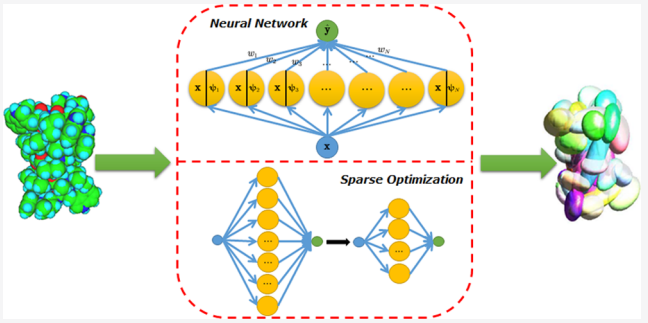
Molecular Sparse Representation by a 3D Ellipsoid Radial Basis Function Neural Network via L1 Regularization

Sheng Gui, Zhaodi Chen, Benzhuo Lu,* and Minxin Chen*

Cite This: J. Chem. Inf. Model. 2020, 60, 6054−6064 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: The three-dimensional structures and shapes of biomolecules provide essential information about their interactions and functions. Unfortunately, the computational cost of biomolecular shape representation is an active challenge which increases rapidly as the number of atoms increase. Recent developments in sparse representation and deep learning have shown significant improvements in terms of time and space. A sparse representation of molecular shape is also useful in various other applications, such as molecular structure alignment, docking, and coarse-grained molecular modeling. We have developed an ellipsoid radial basis function neural network (ERBFNN) and an algorithm for sparsely representing molecular shape. To evaluate a sparse representation model of molecular shape, the Gaussian density map of the molecule is approximated using ERBFNN with a relatively small number of neurons. The deep learning models were trained by optimizing a nonlinear loss function with L1 regularization. Experimental results reveal that our algorithm can represent the original molecular shape with a relatively higher accuracy and fewer scale of ERBFNN. Our network in principle is applicable to the multiresolution sparse representation of molecular shape and coarse-grained molecular modeling. Executable files are available at https://github.com/SGUI-LSEC/SparseGaussianMolecule. The program was implemented in PyTorch and was run on Linux.



INTRODUCTION

Recent advances in the field of deep learning and neural networks have gained revolutionary achievements in computer vision and related applications such as object detection,[1] image classification,[2] and semantic segmentation.[3] A radial basis function (RBF) network is a special class of feedforward neural networks (FNNs), which have certain advantages over other types of FNNs, such as simpler network structures and a faster training process. Due to good approximation capabilities, single-output RBF networks are usually utilized to model nonlinear functions in engineering applications. In practice, learning of RBF networks includes two assignments: determining the RBF network structure and optimizing the adaptable parameters (such as centers and the radii of RBF neurons, and linear output weights). Parameter optimization and network construction are two important and closely related issues. Solving these two issues at the same time is a difficult mixed integer problem.[4] Owing to the lack of promising methods to address this integrated problem, the two tasks are solved respectively in many learning algorithms of RBF networks.[5] In this case, the network structure is determined in advance, and the parameters are then trained by algorithms of supervised learning. Typically optimizing the empirical risk leads to an overfitting problem, which causes poor generalization capability. To tackle this issue, regularization techniques, such as L1 and L2 regularization techniques,[6−8] are common components in modern machine learning. Fundamentally, a regularization term is added to the empirical risk to penalize over-complicated solutions. L1 regularization is implemented by adding a weighted L1 norm of the parameter vector to the loss function, which ensures that the sum of the absolute values of the parameters is small, while L2 regularization uses the L2 norm, which makes the sum of the squares of the parameters small. There has been an increasing interest in L1 regularization because of its advantages over L2 regularization.[9] For example, L1 regularization usually produces sparse parameter vectors in which many parameters are close to zero. Thus, more sparse solution can be obtained. In particular, deep learning has significantly improved the sparsity of molecular shape representation and related tasks such as molecular docking, alignment, drug design, and multiscale modeling.

Received: May 26, 2020
Published: November 12, 2020

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

Biomolecules such as proteins are the fundamental functional units of life activities. Geometric modeling of biomolecules plays an important role in the fields of computer-aided drug design and computational biology. In the computer-aided drug design field, biomolecular shape has remained an important research subject for many years, for instance, in shape-based docking problems,[10] molecular shape comparisons,[11] calculating molecular surface areas,[12,13] coarse-grained molecular dynamics,[14] the generalized Born models,[15] etc; and the biomolecular geometric shape (especially molecular surface) is a prerequisite for using the boundary element method (BEM) and finite element method (FEM) in implicit solvent models.[16]

Due to high complexity and irregularity of the large size molecular models, new issues arise in simulations and other downstream applications.[17] Therefore, efficient representation of the molecular shape (as well as the "molecular surface" or "molecular volume") for large size biomolecules with high quality is an open challenge.[16]

The molecular shape is defined in various ways.[18] For molecular volume, the Gaussian density map is a suitable representation of the molecular shape, as its density maps provide a realistic representation of the volumetric synthetic electron density maps for the biomolecules. With respect to molecular surfaces, there are four important molecular surfaces: van der Waals (VDW) surface, solvent accessible surface (SAS),[19] solvent excluded surface (SES),[20] and Gaussian surface. The van der Waals surface is the smallest envelope enclosing a collection of spheres representing all the atoms in the system with their van der Waals radii. The SAS[19] is the trace of the centers of probe spheres rolling over the van der Waals surface. The SES[20] is the surface traced by the inward-facing surface of the probe sphere. The Gaussian surface[13,21] is a level set of the summation of the spherically symmetrical volumetric Gaussian density distribution centered at each atom of the biomolecular system. In 2015, Liu et al. presented that the VDW surface, SAS, and SES can be approximated well by the Gaussian surface with proper parameter selection.[12] Compared with VDW surface, SAS and SES, the Gaussian surface is smooth and has been widely used in many problems in computational biology.[10,11,13−15] Thus, in this paper, we adopt an ellipsoid RBF neural network to approximate the Gaussian density maps of the molecular shape. The Gaussian density maps and the Gaussian surface descriptions of the specific forms will be discussed in the next section.

For Gaussian density maps, the volume Gaussian function is constructed by a summation of Gaussian kernel functions, whose number depends on the total number of atoms in the molecule. Thus, the computational cost for biomolecular surface construction increases as the atom number (number of Gaussian kernel functions) increases. This leads to a significant challenge for their analysis and recognition. In the case of large biomolecules, the number of kernels in their definition of Gaussian molecular surface may reach millions. In 2015, Zhang et al.[22] put forward an atom simplification method for the biomolecular structure based on the Gaussian molecular surface. This method contains two main steps. The first step eliminates low-contributing atoms. The second step optimizes the center location, the radius and the decay rate of the remaining atoms based on the gradient flow method.

In the area of computer-aided geometric design, the Gaussian surface is a classical implicit surface representing method. In the last two decades, the implicit surface reconstruction has gained a key attention of the researchers. For example, Carr et al.[23] proposed a method to reconstruct an implicit surface with RBFs and performed a greedy algorithm to append centers with large residuals to decrease the number of basis functions. However, the result of this method is not sparse enough. Samozino et al.[24] presented a strategy to put the RBF centers on the Voronoi vertices. This strategy first picks a user-specified number of centers by filtering and clustering from a subset of Voronoi vertices, and then gets the reconstructed surface by solving a least-squares problem. However, it leads to a larger approximation error on the surface while approximating the surface and center points equally. In 2016, Li et al.[25] proposed a model of sparse RBF surface representations. They constructed the implicit surface based on sparse optimization with RBF. The initial Gaussian RBF is on the medial axis of the input model. They have solved the RBF surface by sparse optimization technique. Sparse optimization has become a very popular technique in many active fields, for instance, signal processing, computer vision, etc.[26] This technique has been applied in linear regression,[27] deconvolution,[28] signal modeling,[29] preconditioning,[30] machine learning,[31] denoising,[32] and regularization.[33] In the last few years, sparse optimization has also been applied in geometric modeling and graphics problems.[34]

In this paper, based on the structure of the RBF neural network, we propose an ellipsoid RBF neural network for reducing the number of kernels in the definition of the Gaussian surface while preserving the shape of the molecular surface. We highlight several differences and main contributions between our method and previous L1 optimization methods with shape representation:

1. Our focus is mainly on reducing the number of kernels in Gaussian density maps by pruning useless ellipsoid RBF neuron through L1 regularization. Our method uses fewer number of kernels in Gaussian density maps as compared to the state-of-the-art methods.
2. The loss function of our model is a complicated nonlinear function with respect to the locations, sizes, shapes, and orientations of RBFs.
3. Different initializations and training network algorithms are proposed for solving the corresponding optimization problem in our model.

The rest of this paper is organized as follows. Section "Methods" reviews some preliminary knowledge about volumetric electron density maps, Gaussian surface, ellipsoid Gaussian RBF, and ellipsoid RBF network and then presents our model together with an algorithm for representing the Gaussian density maps sparsely. The experimental results and comparisons are demonstrated in section "Results and Discussion". Finally, the paper is concluded in section "Conclusions".

METHODS

Brief Review of Volumetric Electron Density Maps, Gaussian Surface, Ellipsoid Gaussian RBF, and Ellipsoid RBF Network. *Volumetric Electron Density Maps.* Volumetric electron density maps[35−37] are often modelled as volumetric Gaussian density maps $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}$. The definition of the volumetric Gaussian density maps is as follows

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} e^{-d(\|\mathbf{x}-\mathbf{x}_i\|^2 - r_i^2)} \quad (1)$$

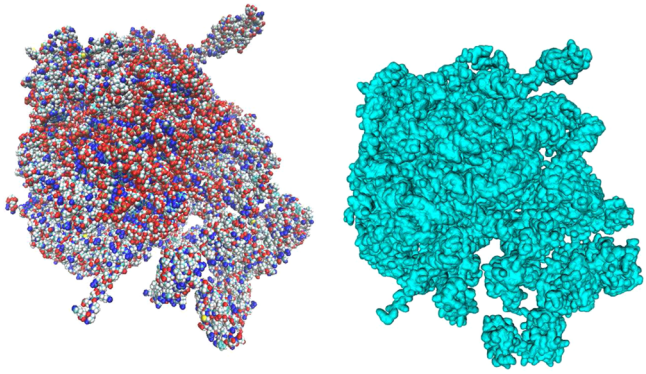
https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

where the parameter d is positive and controls the decay rate of the kernel functions, $\mathbf{x}_i$ and r_i are the location and radius of atom i.

Gaussian Surface. The Gaussian surface is defined as a level set from volumetric synthetic electron density maps

$$\{\mathbf{x} \in \mathbb{R}^3, \phi(\mathbf{x}) = c\} \tag{2}$$

where c is the isovalue, and it controls the volume enclosed by the Gaussian density maps. Figure 1 shows an example of a



(a) (b)

Figure 1. Example of Gaussian molecular surface via VCMM.[38] (a) The VDW surface and (b) the Gaussian molecular surface generated using TMSmesh.[39−41] with parameter d and c being set as 0.9 and 1.0, respectively. All coordinates and corresponding radii are drawn from the PQR file that is transformed from the PDB file, using the PDB2PQR tool.[42]

Gaussian surface. This molecule contains the entire 70S ribosome, including the 30S subunit (16S rRNA and small subunit proteins), 50S subunit (23S rRNA, 5S rRNA, and large subunit proteins), P- and E-site tRNA, and messenger RNA. This molecule is obtained from 70S ribosome3.7A model140.pdb.gz on http://rna.ucsc.edu/rnacenter/ribosomedownloads.html. Figure 1a shows all the atoms in the molecule, and Figure 1b shows the corresponding Gaussian surface.

Ellipsoid Gaussian RBF. The RBF is written as $\xi_i(\mathbf{x}) = \xi(\|\mathbf{x} - \mathbf{c}_i\|)$, where $\xi(\mathbf{x})$ is a nonnegative function defined on $[0, \infty)$, $\mathbf{c}_i$ is center location of the ith basis function. RBF has basic properties as follows, $\xi(0) = 1$ and $\lim_{\mathbf{x}\to+\infty}\xi(\mathbf{x}) = 0$. A typical choice of RBF is Gaussian function

$$\xi(\mathbf{x}) = e^{-\mathbf{x}^2} \tag{3}$$

In addition, there are other RBFs including thin plate spline RBF, e.g., $\xi(r) = r^2 \ln(r)$ for $r \in \mathbb{R}$.

Compared with other RBFs, we put forward ellipsoid RBF with parameters with respect to the locations, sizes, shapes, and orientations. The ellipsoid Gaussian RBF can be written as

$$\psi(\mathbf{x}) = e^{-\|\mathbf{D}^{1/2}\boldsymbol{\Theta}(\alpha,\beta,\gamma)(\mathbf{x}-\mathbf{c})\|^2} \tag{4}$$

where $\mathbf{c} = (c_1, c_2, c_3)^T \in \mathbb{R}^3$ is the center of the ellipsoid Gaussian RBF, $\mathbf{D} = \mathrm{diag}(d_1, d_2, d_3)$, where d_i, $i = 1, 2, 3$ defines the length of ellipsoid along the three main axes, $\boldsymbol{\Theta}(\alpha, \beta, \gamma)$ is the total rotation matrix, and it is equal to the product of rotation matrices from three directions

$$\boldsymbol{\Theta}(\alpha, \beta, \gamma) = \boldsymbol{\Theta}_z(\gamma)\cdot\boldsymbol{\Theta}_y(\beta)\cdot\boldsymbol{\Theta}_x(\alpha) \tag{5}$$

and $\boldsymbol{\Theta}_x(\alpha)$ is a rotation matrix of the x direction

$$\boldsymbol{\Theta}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{6}$$

$\boldsymbol{\Theta}_y(\beta)$ is a rotation matrix of the y direction

$$\boldsymbol{\Theta}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{7}$$

$\boldsymbol{\Theta}_z(\gamma)$ is a rotation matrix of the z direction

$$\boldsymbol{\Theta}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

so that $\boldsymbol{\Theta}(\alpha, \beta, \gamma)$ is equal to

$$\begin{bmatrix} \cos\beta\cos\gamma & -\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma + \cos\alpha\cos\gamma\sin\beta \\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma \\ -\sin\beta & \cos\beta\sin\alpha & \cos\alpha\cos\beta \end{bmatrix} \tag{9}$$

Ellipsoid RBF Networks. The RBF network (Figure 2) is a special FNN consisting of three layers:

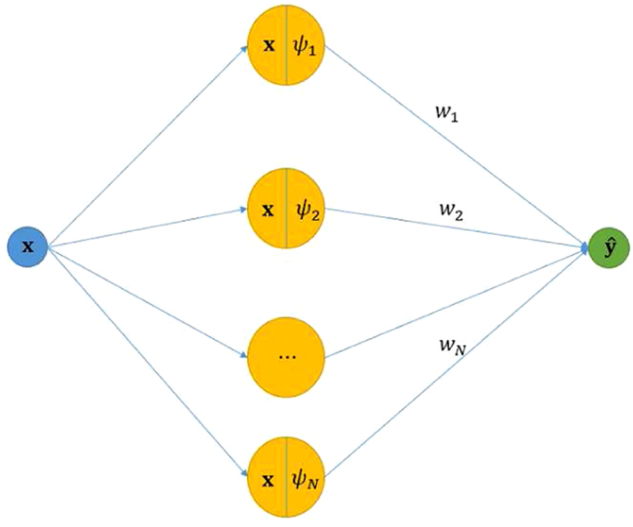


Figure 2. Structure of the ellipsoid RBF neural network.

- an input layer
- a hidden layer with a nonlinear activation function
- a linear output layer

The choice of activation function is the ellipsoid Gaussian function. For an input $\mathbf{x} \in \mathbb{R}^3$, the output of the ellipsoid RBF network is calculated by

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

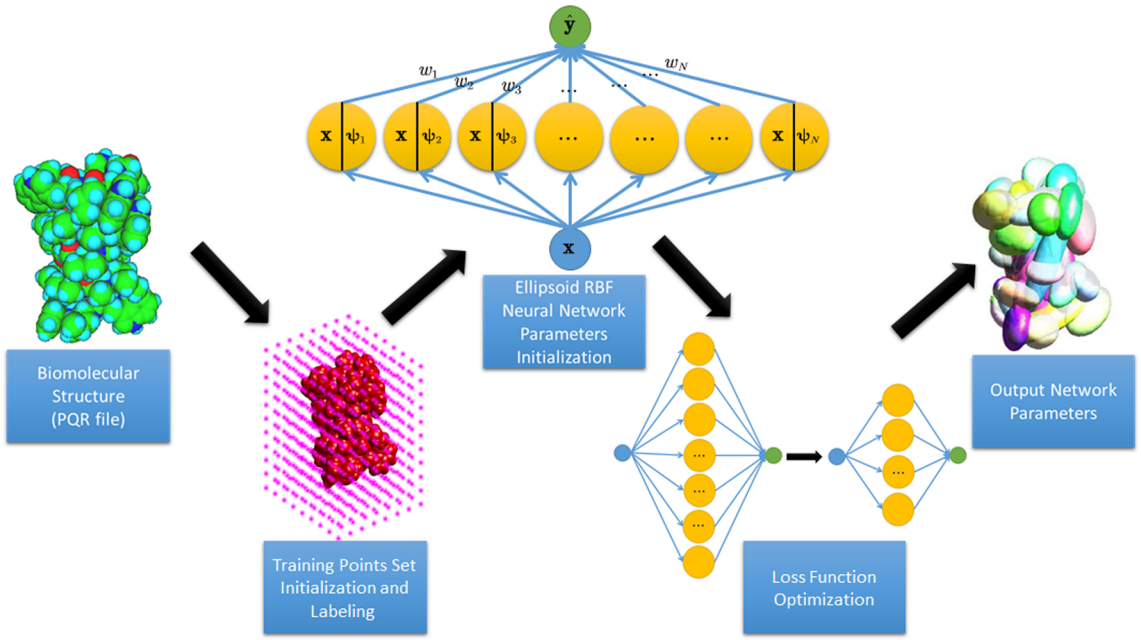


Figure 3. Pipeline of our algorithm and results of each step.

$$\Psi(\mathbf{x}) = \sum_{i=1}^{N} w_i \psi_i(\mathbf{x}) = \sum_{i=1}^{N} w_i e^{-\|\mathbf{D}_i^{1/2}\boldsymbol{\Theta}_i(\alpha_i,\beta_i,\gamma_i)(\mathbf{x}-\mathbf{c}_i)\|^2} \tag{10}$$

where $\mathbf{c}_i = [c_{i1}, c_{i2}, c_{i3}]^T \in \mathbb{R}^3$ is the ith ellipsoid RBF center of the hidden layer, $\mathbf{D}_i = \mathrm{diag}(d_{i1}, d_{i2}, d_{i3})$ represents the lengths of corresponding ellipsoid RBF along three main axes of the hidden layer, $\boldsymbol{\Theta}_i(\alpha_i, \beta_i, \gamma_i)$ is a rotation matrix of the ith neuron. w_i is the output weight between the ith hidden neuron and the output node. And $\|\cdot\|$ is the L2 norm of the vector.

The parameters (i.e., the weights connecting the neuron to the output layer, lengths of ellipsoid RBF along three main axes, center coordinates, and rotation angles) of the hidden neuron are denoted by $\boldsymbol{\sigma} = [\mathbf{w}, \mathbf{d}, \mathbf{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}]^T \in \mathbb{R}^{10N}$. The descriptions of the specific forms of $\boldsymbol{\sigma}$ will be given in the following section. Assume that the training data set is given by $\{(\mathbf{x}_m, \mathbf{y}_m)|\mathbf{x}_m \in \mathbb{R}^3, \mathbf{y}_m \in \mathbb{R}, m = 1, 2, \cdots, M\}$, where $\mathbf{x}_m$ is the mth inputs and $\mathbf{y}_m$ is the desired output value for the mth inputs. The actual output vector can be calculated by

$$\hat{\mathbf{Y}} = \mathbf{H}\mathbf{w} \tag{11}$$

where

$$\mathbf{H} = \begin{bmatrix} e^{-\left\|\mathbf{D}_1^{1/2}\boldsymbol{\Theta}_1(\mathbf{x}_1-\mathbf{c}_1)\right\|_2^2} & e^{-\left\|\mathbf{D}_2^{1/2}\boldsymbol{\Theta}_2(\mathbf{x}_1-\mathbf{c}_2)\right\|_2^2} & \cdots & e^{-\left\|\mathbf{D}_N^{1/2}\boldsymbol{\Theta}_N(\mathbf{x}_1-\mathbf{c}_N)\right\|_2^2} \\ e^{-\left\|\mathbf{D}_1^{1/2}\boldsymbol{\Theta}_1(\mathbf{x}_2-\mathbf{c}_1)\right\|_2^2} & e^{-\left\|\mathbf{D}_2^{1/2}\boldsymbol{\Theta}_2(\mathbf{x}_2-\mathbf{c}_2)\right\|_2^2} & \cdots & e^{-\left\|\mathbf{D}_N^{1/2}\boldsymbol{\Theta}_N(\mathbf{x}_2-\mathbf{c}_N)\right\|_2^2} \\ \vdots & \vdots & & \vdots \\ e^{-\left\|\mathbf{D}_1^{1/2}\boldsymbol{\Theta}_1(\mathbf{x}_M-\mathbf{c}_1)\right\|_2^2} & e^{-\left\|\mathbf{D}_2^{1/2}\boldsymbol{\Theta}_2(\mathbf{x}_M-\mathbf{c}_2)\right\|_2^2} & \cdots & e^{-\left\|\mathbf{D}_N^{1/2}\boldsymbol{\Theta}_N(\mathbf{x}_M-\mathbf{c}_N)\right\|_2^2} \end{bmatrix}_{M\times N} \tag{12}$$

$\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, ..., \hat{\mathbf{y}}_M]^T$ is an output value vector for M inputs, $\mathbf{w} = [w_1, w_2, ..., w_N]^T$ is a N-dimensional vector, and w_k is the weight connecting the kth hidden neuron to the output layer. The error vector is defined as

$$\mathbf{e} = [e_1, e_2, \cdots, e_N]^T \tag{13}$$

with $e_i = \hat{y}_i - y_i$

Model and Algorithm. *Modeling with an Ellipsoid RBF Network.* The major goal of this study is to create a sparse representation of the Gaussian molecular model by the ellipsoid RBF neural network. According to the definition of volumetric electron density maps and structure of the ellipsoid RBF network, the loss function for sparsely representing a Gaussian molecular model is as follows

$$\mathcal{L}(\boldsymbol{\sigma}) = \rho_1 \cdot [\|\mathbf{w}\|_1 + \|\mathbf{d}\|_1] + \rho_2 \cdot E_s(\boldsymbol{\sigma}) \tag{14}$$

and corresponding constrained condition is

$$\mathbf{w} \geq 0, \mathbf{d} \geq 0, \mathbf{w} \in \mathbb{R}^N, \mathbf{d} \in \mathbb{R}^{3N} \tag{15}$$

The first term in eq 14 is a L1 regularization term to reduce both network complexity and overfitting. The formulate is as follows

$$\|\mathbf{w}\|_1 + \|\mathbf{d}\|_1 = \sum_{i=1}^{N} |w_i| + \sum_{i=1}^{N}\sum_{j=1}^{3} |d_{ij}| \tag{16}$$

where $\mathbf{w} = [w_1, w_2, ..., w_N]$, $\mathbf{d} = [d_{11}, d_{12}, d_{13}, ..., d_{N1}, d_{N2}, d_{N3}]$.

The second $E_s(\boldsymbol{\sigma})$ is density error between the sparsely represented molecule and original molecule at the training points set $\mathbf{x}_m$, $m = 1, 2, ..., M$. We have

$$\begin{aligned} E_s(\boldsymbol{\sigma}) &= \sum_{m=1}^{M} [\Psi(\mathbf{x}_m; \boldsymbol{\sigma}) - \phi(\mathbf{x}_m)]^2 \\ &= \sum_{m=1}^{M} \left[\sum_{i=1}^{N} w_i e^{-\|\mathbf{D}_i^{1/2}\boldsymbol{\Theta}_i(\mathbf{x}_m-\mathbf{c}_i)\|_2^2} - \phi(\mathbf{x}_m) \right]^2 \end{aligned} \tag{17}$$

where $\Psi(\mathbf{x})$ is an ellipsoid RBF neuron network. $\phi(\mathbf{x})$ is the volumetric electron density map (eq 1) and is approximated by $\Psi(\mathbf{x})$. $\mathbf{x}_m = (x_{m1}, x_{m2}, x_{m3})^T \in \mathbb{R}^3$ is the coordinates of the mth training point. $\mathbf{c}_i = (c_{i1}, c_{i2}, c_{i3})^T \in \mathbb{R}^3$ is the center of the ith activation function of the ellipsoid RBF. $\mathbf{D}_i = \mathrm{diag}(d_{i1}, d_{i2}, d_{i3})$ defines the lengths of the ellipsoid along three main axes. $\boldsymbol{\Theta}_i$ is a rotation matrix. $\alpha_i, \beta_i, \gamma_i$ are rotation angles of the ith activation function of the ellipsoid RBF neuron, $i = 1, 2, ..., N$ and $m = 1, 2,$

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

..., M. N is the number of the ellipsoid RBF neurons. M is the number of training points. $\boldsymbol{\sigma}$ is the network parameter

$$\boldsymbol{\sigma} = [\mathbf{w}, \mathbf{d}, \mathbf{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}]^{\mathrm{T}} \tag{18}$$

where $\mathbf{w} = [w_1, w_2, ..., w_N]$, $\mathbf{d} = [d_{11}, d_{12}, d_{13}, ..., d_{N1}, d_{N2}, d_{N3}]$, $\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N]$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_N]$, $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_N]$, and $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_N]$.

The two parameters $\rho_1 > 0$ and $\rho_2 > 0$ are used to balance the two targets, accuracy (E_s) and sparsity (L1-regularization), and the constrained conditions are explained as follows, (i) $\mathbf{w} \geq 0$ indicates that the corresponding ellipsoid Gaussian RBF is nonnegative. (ii) $\mathbf{d} \geq 0$ implies that the activation functions do not blow up at infinity, which is consistent with the fitted function ϕ. In order to transform the eqs 14 and 15 to an unconstrained loss function, we perform the following substitution,

$$\begin{cases} w_i = \tilde{w}_i^2, \\ d_{iq} = \tilde{d}_{iq}^2, \\ i = 1, 2, \cdots, N, \ q = 1, 2, 3 \end{cases} \tag{19}$$

and corresponding $\tilde{\mathbf{D}}_i = \mathrm{diag}(\tilde{d}_{i1}^2, \tilde{d}_{i2}^2, \tilde{d}_{i3}^2)$. For simplicity, we still use w_i, d_{iq}, $\mathbf{D}_i$ to denote $\tilde{w}_i, \tilde{d}_{iq}$, $\tilde{\mathbf{D}}_i$.

Thus, the loss function of the ellipsoid RBF network for sparsely representing a molecule is

$$\mathcal{L}(\boldsymbol{\sigma}) = \rho_1 \cdot [\|\mathbf{w}\|_1 + \|\mathbf{d}\|_1] + \rho_2 \cdot \sum_{m=1}^{M} \left[\sum_{i=1}^{N} w_i^2 \cdot \mathrm{e}^{-\|\mathbf{D}_i \boldsymbol{\Theta}_i (\mathbf{x}_m - \mathbf{c}_i)\|_2^2} - \phi(\mathbf{x}_m) \right]^2 \tag{20}$$

Overview. In this section, we describe the algorithms to construct sparse representation of Gaussian molecular density by the ellipsoid RBF neural network. The inputs of our method are PQR files which include a list of centers and radii of atoms. The output of our method is network parameters which contain the centers, the lengths, the rotation angles of the ellipsoid RBF neural network, and the weights connecting the hidden neurons to the output layer. The algorithm outline is as follows: first, set the training points $\mathbf{x}_m$, $m = 1,...,M$ and label corresponding value $\phi(\mathbf{x}_m)$. Second, initialize the ellipsoid RBF network (i.e., the number of neurons, the parameters of the ellipsoid RBF neural network). Third, optimize the loss function in eq 20 using an ADAM algorithm[43] to minimize the sparsity and error terms in eq 20 alternatively. Figure 3 illustrates the process of our algorithm. The result shows that using our method, the original Gaussian surface is approximated well by a summation of much fewer ellipsoid Gaussian RBFs.

Training Points Set Initialization and Labeling. In order to train a network, in the first step, the training points set is initialized. The molecule is put in a bounding box Ω (Figure 4a) in $\mathbb{R}^3$. The range of bounding box is $[a, b] \times [c, d] \times [e, f]$, where $a, b, c, d, e, f \in \mathbb{R}$. The bounding box Ω is discretized into a set of uniform grid as shown in Figure 4b. The training points are the grid points defined as follows

$$\{\mathcal{P}_{ijk}\} = \{(x_i, y_j, z_k) | x_i = a + i \cdot (b - a)/N_x, y_j = c + j \cdot (d - c)/N_y, z_k = e + k \cdot (f - e)/N_z\} \tag{21}$$

where $i = 0, 1, 2, ..., N_x$, $j = 0, 1, 2, ..., N_y$, and $k = 0, 1, 2, ..., N_z$. N_x, N_y, and N_z are the total number of indices i, j, and k, respectively.

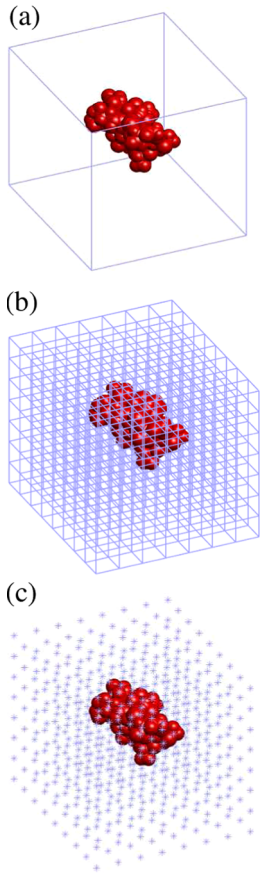


Figure 4. Training points set initialization. (a) A real molecule (PDBID: 1GNA) within a bounding box. (b) A set of uniform grid of the bounding box. (c) Initial training points.

In the second step, the points $\{\mathcal{P}_{ijk}\}$ of the training set is labeled for training network parameter. Label of $\{\mathcal{P}_{ijk}\}$ is calculated in the following form: $\{\mathcal{P}_{ijk}\}_{\mathrm{label}} = \phi(\{\mathcal{P}_{ijk}\})$. A set of training points $\{\mathbf{x}_m\}_{m=1}^M$ is chosen from the set of uniform grid points $\{\mathcal{P}_{ijk}\}$. To achieve good preservation of the molecular shape the selected points $\{\mathbf{x}_m\}_{m=1}^M$ are close to the Gaussian surface defined in eq 2. In this paper, the training points set $\{\mathbf{x}_m\}_{m=1}^M$ satisfying $\|\phi(\mathbf{x}_m) - c\|_2 \leq 1$ are selected.

Parameter Initialization of the Ellipsoid RBF Neural Network. In this section, we initialize the ellipsoid RBF neural network parameters $\boldsymbol{\sigma}$ defined in eq 18. Considering that the Gaussian RBF is a degradation case of the ellipsoid Gaussian RBF, the activation function ψ can be initialized in the same manner as ϕ. Thus, the strategy of initialization is as follows,

1. The lengths of ellipsoid RBFs $\mathbf{d}$ are set to be constant vectors. In this paper, the initial $\mathbf{d}$ can be set as follows

$$\mathbf{d} = [0.5, 0.5, \cdots, 0.5]^{\mathrm{T}} \in \mathbb{R}^{3N} \tag{22}$$

where N is the number of atoms.

2. The initial angles of ellipsoid RBFs $\boldsymbol{\Theta}$ are set to be zeros

$$\boldsymbol{\Theta} = \mathbf{0} \in \mathbb{R}^{3N} \tag{23}$$

3. The initial center coordinates $\mathbf{c}$ of ellipsoid RBFs are given by the centers of atoms as follows

$$\mathbf{c}_i = [x_{\mathrm{atom}}^{(i)}, y_{\mathrm{atom}}^{(i)}, z_{\mathrm{atom}}^{(i)}]^{\mathrm{T}}, \quad i = 1, 2, \cdots, N \tag{24}$$

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

where $x_{atom}^{(i)}$, $y_{atom}^{(i)}$, and $z_{atom}^{(i)}$ are coordinates of the ith atom.

4. While $\mathbf{d}$, $\mathbf{c}_i$, and $\mathbf{\Theta}_i$ have been chosen and atom radii $r = [r_1, r_2, ..., r_N]$ are given, to initialize ellipsoid RBF activation function ψ similar to RBF ϕ, we set the weight $\mathbf{w}$ of the ellipsoid Gaussian RBF as follows

$$\mathbf{w} = [e^{r_1^2/4}, e^{r_2^2/4}, \cdots, e^{r_N^2/4}]^T \tag{25}$$

Sparse Optimization. After initialization of the ellipsoid RBF neural network, the sparsity of Gaussian RBF representation is computed by minimizing the loss function (eq 20). We aim at finding a sparse weight vector $\mathbf{w} = [w_1, w_2, ..., w_N]^T$ and a sparse decay rate vector $\mathbf{d} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]^T$ of $\psi(\boldsymbol{\sigma})$ as defined in eq 10 such that the ellipsoid RBF network ψ is a good fit of ϕ at the training points set $\{\mathbf{x}_m\}_{m=1}^M$. Algorithm 1 represents the main modules of our sparse optimization method, which is described below.

Algorithm 1 Sparse optimization

Input: PQR file including coordinates of centers and radii of atoms.
Output: The list of parameters of ellipsoid RBF neural network, i.e. solution of $\boldsymbol{\sigma}$ in minimizing $\mathscr{L}(\boldsymbol{\sigma})$.
Step 1. initialize network parameters $\boldsymbol{\sigma}$ as described in this subsection "Parameter initialization of ellipsoid RBF neural network".
Step 2. select training points set $\{\mathbf{x}_m\}_{m=1}^M$ as described in this subsection "Training points set initialization and labelling".
Step 3. set the number of maximum iteration `MaxNiter` and number of sparse optimization iteration `SparseNiter`. set the coefficients ρ_1, ρ_2 in Eq. 14.
Step 4. initialize the variable of iteration: `Niter` $= 0$ and set tolerance: $\texttt{tol}_1$ and $\texttt{tol}_2$.
Step 5. select the size of batch `Batch_size` for optimization algorithm.
Step 6. optimize the two terms of loss function in Eq. 20 alternatively.
while `Niter` $<$ `MaxNiter` **do**
  `Niter` $=$ `Niter` $+ 1$
  Step 6.1. prune the useless the ellipsoid RBF neuron $|w_i| < \texttt{tol}_1$ every $\texttt{check}_{\texttt{step}}$ steps.
  Step 6.2. calculate $\psi(\mathbf{x}_m)$ for training points set.
  Step 6.3. check the maximum of error between ψ and ϕ at training points set $\{\mathbf{x}_m\}_{m=1}^M$ and update the coefficients ρ_1 and ρ_2.
  if $\max_{1\le m\le M} \|\psi(\mathbf{x}_m) - \phi(\mathbf{x}_m)\| > \texttt{tol}_2$ **then**
    $\rho_1 = 1, \rho_2 = 0$
  end if
  Step 6.4. set coefficients ρ_1 and ρ_2 for accuracy optimization.
  if `Niter` $>$ `SparseNiter` **then**
    $\rho_1 = 1, \rho_2 = 0$
  end if
  Step 6.5. optimize the loss function $\mathscr{L}(\boldsymbol{\sigma})$ by batch ADAM algorithm.
end while

Step 1 shows the initialization of parameters $\boldsymbol{\sigma}$ for the ellipsoid RBF neural network. Step 2 selects training points set $\{\mathbf{x}_m\}_{m=1}^M$. Step 3 and step 4 initialize some variables, i.e., the number of maximum iterations, the number of sparse optimization iterations, error tolerance, and the coefficients ρ_1 and ρ_2. Step 5 sets the size of the batch ($\texttt{Batch\_size} = \min\{M, 10\,000\}$) for the optimization algorithm. Step 6 shows the numerical algorithm of optimization for the loss function (eq 20). Step 6.1 prunes the ellipsoid RBF neuron useless if the corresponding weight w_i connecting the ith hidden neuron to the output layer is less than $\texttt{tol}_1$ per $\texttt{check}_{\texttt{step}}$ steps. In this paper, we set $\texttt{tol}_1 = 1e-3$ and $\texttt{check}_{\texttt{step}} = 20$. Step 6.2 calculates the prediction value ψ for all training points set. Step 6.3 checks the maximum of error between ψ and ϕ at training points set $\{\mathbf{x}_m\}_{m=1}^M$ and updates the coefficients ρ_1 and $\rho_2(\texttt{tol}_2 = 0.1)$. Step 6.4, after performing `SparseNiter` iterations, with the number of effective neurons fixed, keeps performing some steps of minimization of E_s to achieve better accuracy of the approximation on training points set. Step 6.5 updates the network parameter $\boldsymbol{\sigma}$ and optimizes loss function $\mathcal{L}(\boldsymbol{\sigma})$ by the batch ADAM[43] method. The pipeline of step 6.5 is as follows

$$m_k = \beta_1 \cdot m_{k-1} + (1 - \beta_1) \cdot \nabla \mathcal{L}_k$$

$$v_k = \beta_2 \cdot v_{k-1} + (1 - \beta_2) \cdot \nabla (\mathcal{L}_k)^2$$

$$\boldsymbol{\sigma}_k = \boldsymbol{\sigma}_{k-1} - \tau \times \frac{\frac{m_k}{1-\beta_1^k}}{\sqrt{\frac{v_k}{1-\beta_2^k}} + \epsilon}$$

where $\tau = 0.002$ is the learning rate. β_1, β_2, and ϵ are set as default values ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$), m_k is the kth biased first moment estimate ($m_0 = 0$). v_k is the kth biased second raw estimate ($v_0 = 0$).

RESULTS AND DISCUSSION

In this section, we present some numerical experimental examples to illustrate the effectiveness of our network and methods for representing the Gaussian surface sparsely. Comparisons are made among our network, the original definition of Gaussian density maps and sparse RBF methods.[25] A set of biomolecules taken from the RCSB Protein Data Bank is chosen as a benchmark set. The number of atoms in these biomolecules ranges from hundreds to thousands. These molecules are chosen randomly from RCSB Protein Data Bank, and no particular structure is specified. The implementation of the algorithms is based on PyTorch.[44] All computations are run on an Nvidia Tesla 1080Ti GPU. Further quantitative analysis of the result is given in the following subsections.

Sparse Optimization Results. Twenty biomolecules are chosen to be sparsely represented by our ellipsoid RBF neural network and the sparse RBF method.[25] The Sparse RBF is a sparse RBF surface representation method for a general implicit surface. The process of the Sparse RBF method is as follows. First, the centers of RBF are set on the medial axis of the input surface mesh. Second, the surface points and offset points are set according to the input surface. Finally, the ADMM algorithm[45] is used to solve the corresponding L1 minimization problem. We emphasize several differences between our method and the sparse RBF method: (1) Our focus is mainly on reducing the number of kernels in Gaussian density maps (eq 1) for biomolecular complexes, and sparse RBF method focuses on sparsely representing surface meshes; (2) The input of the Sparse RBF method is a point cloud from the surface mesh. The input of our method is a set of constrained grid points close to the Gaussian surface; (3) Different initializations are proposed for solving the corresponding sparse optimization problem in our model. Base on analytical Gaussian density maps, the locations of our initial RBFs are given by the centers of atoms. For the sparse RBF method, the coordinates of initial RBFs are set by the medial axis of surface; (4) The objective function of the sparse RBF method is a linear function with respect to the combination coefficients of RBFs, and the objective function of our model is a complicated nonlinear function with respect to the combination coefficients, locations, sizes, shapes, and orientations of RBFs. So our model is a nonlinear optimization problem with contributions, locations, orientations, and shapes of basis, since all these parameters are optimizable. This leads to much sparser results. Table 1 shows the final number of effective basis functions from the results of our method and the sparse RBF method.

Figure 5 presents the relationship between the number of ellipsoid RBF neurons in final sparse representation and the number of atoms in the corresponding molecule. The number of atoms for the original Gaussian molecular density map is shown

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

Table 1. Sparse Optimization Results for 20 Test Protein Molecules[a]

INDEX	PDBID	NATOM	Sparse RBF	our method
1	ADP	39	13	5
2	2LWC	75	51	6
3	3SGS	94	56	9
4	1GNA	163	108	17
5	1V4Z	266	266	22
6	1BTQ	307	252	25
7	2FLY	355	267	28
8	6BST	478	315	49
9	1MAG	552	502	46
10	2JM0	569	424	52
11	1BWX	643	537	54
12	2O3M	714	566	62
13	FAS2	906	722	76
14	2IJI	929	742	72
15	3SJ4	1283	953	132
16	3LOD	2315	1810	180
17	1RMP	3514	2871	271
18	1IF4	4251	3288	301
19	1BL8	5892	3491	452
20	AChE	8280	4438	748

[a]NATOM is the number of atoms for each molecule. The fourth column shows the number of RBFs by the Sparse RBF method. The last column shows the number of the ellipsoid RBF neural network. The decay rate d in eq 1 is uniformly taken as 0.5.

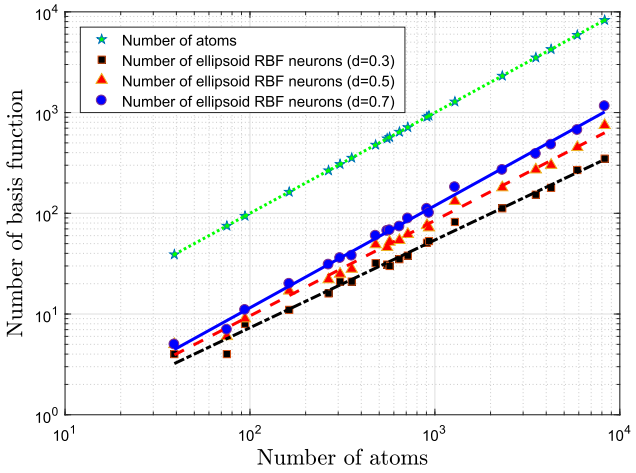


Figure 5. Relationship between the number of atoms and the number of ellipsoid RBF neurons after sparse representation.

by green lines with pentagram markers. To present sparsity of final results from our method, we define the sparse ratio R_s as: $R_s = \frac{N_{ERBF}}{N_{ATOM}}$, where N_{ERBF} is the number of ellipsoid RBF neurons and N_{ATOM} presents the number of atoms. In Figure 5, the changes of sparse ratios with respect to number of atoms for different decay rates (d in eq 1 equals to 0.3, 0.5 and 0.7) are shown by solid lines with square, triangle, and circle markers. The results show that the larger the decay rate d (leading to more rugged and complex molecular surface), the bigger the sparse ratio is going to be. The sparse ratios for the Gaussian molecular density map with $d = 0.3$ are smaller than those of Gaussian molecular density map with $d = 0.7$ as shown in Figure 5. Table 2 shows the sparse ratio for 20 test proteins. The maximum sparse ratio is 0.1419. The minimum sparse ratio is 0.0421.

Figure 6 shows the loss function and the number of ellipsoid RBF neurons decrease as the number of iterations increases in the sparse optimization procedure for molecule 1MAG. In this experiment, the `MaxNiter` and `SparseNiter` are set to be 10 000 and 6000, respectively. After 6000 iterations, ρ_1 is set at zero to minimize E_s term solely, thus the value of the loss function has an abrupt change. The number of ellipsoid RBF neurons decreases dramatically during the iteration process. As shown in Figure 6, the model with 46 ellipsoid RBF neurons achieves minimum error with a relatively less number of ellipsoid RBF neurons.

The computation time of training algorithm is shown in Table 3 for our network. From our experiments, the runtime is approximately proportional to the number of atoms.

Shape Preservation and Further Result Analysis. In this subsection, we first check whether the Gaussian surface is preserved after the process of sparse representation through our method. The area, the volume enclosed by the surface, and the Hausdorff distance are the three criteria to judge whether two surfaces are close enough. These criteria can be calculated on the triangular mesh of the surface. The triangular meshes of molecular surfaces before and after sparse representation are generated through `isosurface` function in MATLAB. For a triangular surface mesh, the surface area S is determined using the following equation

$$S = \frac{1}{2}\sum_{i=1}^{n_f}\left\|\overrightarrow{V_1^iV_2^i} \times \overrightarrow{V_1^iV_3^i}\right\| \tag{26}$$

where n_f is the number of triangle elements and V_1^i, V_2^i, and V_3^i denote the coordinates of the three vertices for the ith triangle.

The volume V enclosed by the surface mesh is determined using the following equation

$$V = \frac{1}{6}\sum_{i=1}^{n_f}\overrightarrow{V_2^iV_1^i} \times \overrightarrow{V_3^iV_1^i}\cdot\vec{c_i} \tag{27}$$

where c_i is the vector from the center of the ith triangle to the origin.

The relative errors of area/volume and the Hausdorff distance are used to characterize the difference between the surfaces before and after sparse representation. The relative errors of area and volume are calculated using the following formulas

$$\text{error}_A = \frac{|A_{our} - A_{original}|}{A_{original}} \tag{28}$$

$$\text{error}_V = \frac{|V_{our} - V_{original}|}{V_{original}} \tag{29}$$

where $A_{original}$ and A_{our} denote the surface areas of the original and our sparsely represented surfaces respectively. $V_{original}$ and V_{our} denote the corresponding volumes enclosed by the original and our surfaces respectively.

The Hausdorff distance between two surface meshes is defined as follows

$$H(S_1, S_2) = \max(\max_{p\in S_1} e(p, S_2), \max_{p\in S_2} e(p, S_1)) \tag{30}$$

where

$$e(p, S) = \min_{p'\in S} d(p, p') \tag{31}$$

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

Table 2. Sparse Ratio for 20 Test Proteins[a]

		$d = 0.3$		$d = 0.5$		$d = 0.7$	
PDBID	NATOM	N_{ERBF}	R_s	N_{ERBF}	R_s	N_{ERBF}	R_s
ADP	39	4	0.1026	5	0.1282	5	0.1282
2LWC	75	4	0.0533	6	0.0800	7	0.0933
3SGS	94	8	0.0851	9	0.0957	11	0.1170
1GNA	163	11	0.0675	17	0.1043	20	0.1227
1V4Z	266	16	0.0602	22	0.0827	31	0.1165
1BTQ	307	21	0.0684	25	0.0814	36	0.1173
2FLY	355	21	0.0592	28	0.0789	38	0.1070
6BST	478	32	0.0669	49	0.1025	60	0.1255
1MAG	552	31	0.0562	46	0.0833	67	0.1214
2JM0	569	30	0.0527	52	0.0914	68	0.1195
1BWX	643	35	0.0544	54	0.0840	74	0.1151
2O3M	714	38	0.0532	62	0.0868	89	0.1246
FAS2	906	51	0.0563	76	0.0839	111	0.1225
2IJI	929	53	0.0571	72	0.0775	101	0.1087
3SJ4	1283	82	0.0639	132	0.1029	182	0.1419
3LOD	2315	112	0.0484	180	0.0778	270	0.1166
1RMP	3514	153	0.0435	271	0.0771	389	0.1107
1IF4	4251	179	0.0421	301	0.0708	481	0.1131
1BL8	5892	269	0.0457	452	0.0767	672	0.1141
AChE	8280	349	0.0421	748	0.0903	1160	0.1401

[a]The decay rate d in eq 1 equals to 0.3, 0.5 and 0.7.

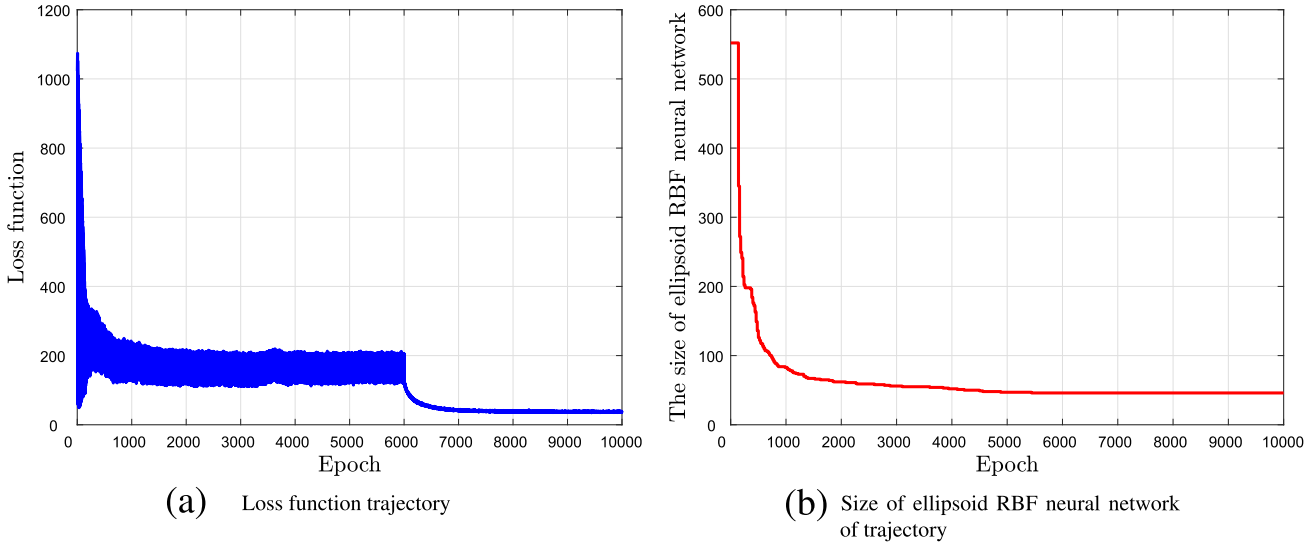


Figure 6. One test of our algorithm on molecule 1MAG. The blue curve is the objective function trajectory during the 10 000 iterations. The red line represents the number of basis functions. The number of initial ellipsoid RBF neurons for this trial is 552 and the number of final ellipsoid RBF neurons is 46.

S_1 and S_2 are two piecewise surfaces spanned by the two corresponding surface meshes, and $d(p, p')$ is the Euclidean distance between the points p and p'. In our work, we use Metro[46] to compute the Hausdorff distance.

The areas and the volumes enclosed by the surfaces before and after the sparse representation for each of the molecules are listed in Table 4. The Hausdorff distances between the original surface and the final surface for the biomolecules are also listed in the last column of Table 4.

Figure 7 shows some visual results of the sparse optimization model. The first column shows original Gaussian surface for five molecules. The second column is the final ellipsoid Gaussian surface in our method, where the blue points represent the location of Gaussian RBF centers. It indicates that our method needs fewer number of ellipsoid RBFs neurons to represent a molecular surface. The third column is the original surface overlapped with the final surface. It shows that the final surface is close to the original surface. The last column shows the configurations of ellipsoid RBF neurons in the sparse representation of five molecules from our method. It demonstrates that after the process of sparse representation, the number of ellipsoid RBF neurons are much sparser than the RBFs. Obviously, each ellipsoid RBF is a local shape descriptor of the molecular shape.

Electrostatic Solvation Energy Calculation Based on the Sparsely Represented Surface. The algorithms introduced in the Methods section are used to generate the sparse surface. We here also test the applicability of the original

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

Table 3. Running Time for 20 Test Proteins[a]

INDEX	PDBID	NATOM	computation time(s)
1	ADP	39	119.773094
2	2LWC	75	118.807773
3	3SGS	94	139.218693
4	1GNA	163	135.889053
5	1V4Z	266	142.258237
6	1BTQ_	307	115.450601
7	2FLY	355	140.081313
8	6BST	478	145.544162
9	1MAG	552	144.182071
10	2JM0	569	119.617039
11	1BWX	643	146.913441
12	2O3M	714	130.578282
13	FAS2	906	156.919386
14	2IJI	929	162.848709
15	3SJ4	1283	273.632593
16	3LOD	2315	375.636324
17	1RMP	3514	608.246697
18	1IF4	4251	795.421310
19	1BL8	5892	1154.25146
20	AChE	8280	2096.61695

[a]The decay rate d in eq 1 is uniformly taken as 0.5.

and the sparse surface in the computations of Poisson−Boltzmann (PB) electrostatics. The boundary element method software used is a publicly available PB solver, AFMPB.[47] Table 5 shows that AFMPB can undergo and produce converged results using the sparse represented surface, and the calculated solvation energies are close to the results using the original surface. Figure 8, using VISIM (www.xyzgate.com), shows the computed electrostatic potentials mapped on the molecular surface. In the future, we can consider adding the charge information to the sparse representation.

CONCLUSIONS

In this paper, a sparse Gaussian molecular shape representation based on ellipsoid RBF neural network is proposed for arbitrary molecules. The original Gaussian density maps is approximated with the ellipsoid RBF neural network. The sparsity of the ellipsoid RBF neural network is computed by solving an L1 regularization optimization problem. Comparisons and experimental results indicate that our network needs much fewer number of ellipsoid RBF neurons to represent the original Gaussian density maps than the isotropic RBF method.

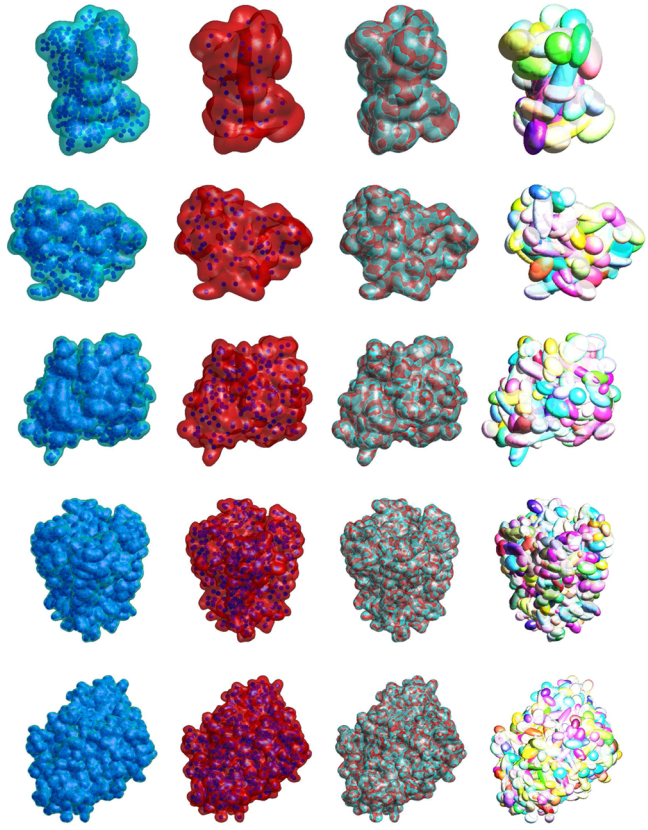


Figure 7. Visual results of our optimization algorithm. Left to right: Original surface (left column), final surface (middle left column), and original surface overlapped with final surface (middle right column), the ellipsoid Gaussian RBFs in the sparse representation from our method (last column). From top to bottom: 1MAG (first row), FAS2 (second row), 3LOD (third row), 1BL8 (fourth row), and AChE (fifth row). The blue points represent the locations of Gaussian RBF centers.

Table 5. Solvation Energy Obtained with the Original Surface and the Sparsely Represented Surface for Five Biomolecules[a]

	solvation energy (kcal/mol)		
molecule	original	sparse	relative error
ADP	−225.992	−230.075	0.0181
2FLY	−238.927	−242.670	0.0157
6BST	−916.715	−920.137	0.0037
2O3M	−3034.82	−3056.04	0.0070
2IJI	−659.502	−665.894	0.0097

[a]Isovalue ϕ = 1.0, initial decay rate d = 0.5.

Table 4. Areas, Volumes, and Hausdorff Distances are Obtained with the Original and the Final Surfaces for 10 Biomolecules[a]

	area (Å^2)			volume (Å^3)			
molecule	original	our	error$_A$	original	our	error$_V$	distance (Å)
ADP	367.9334	358.4047	0.0259	458.0317	454.5578	0.0076	0.6605
2LWC	504.8863	494.5004	0.0206	856.9564	850.5393	0.0075	0.4497
1GNA	1006.1213	995.4826	0.0106	1862.7883	1855.4815	0.0039	0.4764
1BTQ_	1782.1843	1749.4445	0.0184	3412.7345	3406.8652	0.0017	0.6027
1MAG	2479.4398	2438.4246	0.0165	5732.9858	5700.8069	0.0056	0.5441
1BWX	2925.0557	2890.7706	0.0117	6638.2112	6609.3813	0.0043	0.7311
FAS2	3771.6093	3690.4698	0.0215	9198.0803	9168.8722	0.0032	0.7484
2IJI	3783.6502	3731.7199	0.0137	9537.9781	9502.8469	0.0037	0.6187
3SJ4	5887.9106	5797.6176	0.0153	13208.3953	13175.7877	0.0025	0.7209

[a]Isovalue ϕ = 1.0, initial decay rate d = 0.5.

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064

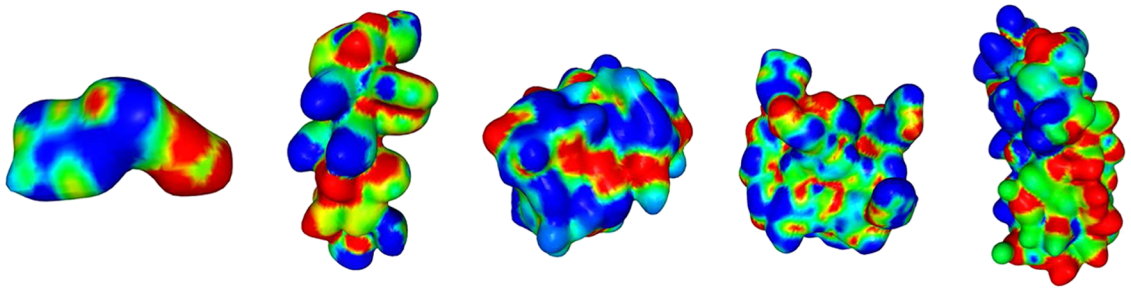


Figure 8. Electrostatic potential of five molecular surfaces, calculated with AFMPB. From left to right: ADP, 2FLY, 6BST, 2O3M, and 2IJI.

For extremely large biomolecular complexes, enough constrained points have to be sampled that are close to the original Gaussian surface. This will lead to a larger optimization problem and will take longer time to solve. As for future work, we will develop more efficient optimization algorithm for our sparse model. Further applications of our model, such as structure alignment and coarse-grained modeling, will also be considered as our future work.

AUTHOR INFORMATION

Corresponding Authors

Benzhuo Lu − *State Key Laboratory of Scientific and Engineering Computing, National Center for Mathematics and Interdisciplinary Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China; School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China;* orcid.org/0000-0003-4159-1532; Email: bzlu@lsec.cc.ac.cn

Minxin Chen − *Department of Mathematics, Soochow University, Suzhou 215006, China*; Email: chenminxin@suda.edu.cn

Authors

Sheng Gui − *State Key Laboratory of Scientific and Engineering Computing, National Center for Mathematics and Interdisciplinary Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China; School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China; Department of Mathematics, Soochow University, Suzhou 215006, China;* orcid.org/0000-0003-0739-9221

Zhaodi Chen − *Department of Mathematics, Soochow University, Suzhou 215006, China*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c00585

Notes

The authors declare no competing financial interest.

All data sets and codes are available on GitHub: https://github.com/SGUI-LSEC/SparseGaussianMolecule. The user needs to install some packages (python3.6, pytorch1.3.1, numpy1.18.0, torchvision0.4.2, and CUDA9.2) before running the code "main.py".

ACKNOWLEDGMENTS

We gratefully acknowledge the anonymous referees for their helpful comments. The authors thank Dr. Khan Dawar, Dr. Wang Qin, and Dr. Shen Ruigang for proofreading and language revision. This work was supported by the National Key Research and Development Program of China (grant 2016YF-B0201304) and the China NSF (NSFC 11771435 and NSFC 21573274).

REFERENCES

(1) Sermanet, P.; Eigen, D.; Zhang, M.; Xiang, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, arXiv:1312.6229. arXiv.org e-Print archive. https://arxiv.org/abs/1312.6229 (submitted Dec 21, 2013).

(2) Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv:1602.07261. arXiv.org e-Print archive. https://arxiv.org/abs/1602.07261 (submitted Feb 23, 2016).

(3) Krähenbühl, P.; Koltun, V. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials https://arxiv.org/abs/1210.5644 (submitted Oct 20, 2012).

(4) Peng, J.; Li, K.; Irwin, G. W. A Novel Continuous Forward Algorithm for RBF Neural Modelling. *IEEE Trans. Autom. Control* **2007**, *52*, 117−122.

(5) Schwenker, F.; Kestler, H.; Palm, G. Three Learning Phases for Radial-Basis-Function Networks. *Neural Networks* **2001**, *14*, 439−458.

(6) Perkins, S.; Lacker, K.; Theiler, J. Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space. *J. Mach. Learn. Res.* **2003**, *3*, 1333−1356.

(7) Shen, C.; Li, H.; van den Hengel, A. Fully Corrective Boosting with Arbitrary Loss and Regularization. *Neural Networks* **2013**, *48*, 44−58.

(8) Vidaurre, D.; Bielza, C.; Larranaga, P. Learning an L1-Regularized Gaussian Bayesian Network in the Equivalence Class Space. *IEEE Trans. Syst., Man, and Cybern., Part B* **2010**, *40*, 1231−1242.

(9) Qian, X.; Huang, H.; Chen, X.; Huang, T. Efficient Construction of Sparse Radial Basis Function Neural Networks Using L1-Regularization. *Neural Networks* **2017**, *94*, 239−254.

(10) McGann, M.; Almond, H.; Nicholls, A.; Grant, J.; Brown, F. Gaussian Docking Functions. *Biopolymers* **2003**, *68*, 76−90.

(11) Grant, J.; Gallardo, M.; Pickup, B. A Fast Method of Molecular Shape Comparison: A Simple Application of a Gaussian Description of Molecular Shape. *J. Comput. Chem.* **1996**, *17*, 1653−1666.

(12) Liu, T.; Chen, M.; Lu, B. Parameterization for Molecular Gaussian Surface and a Comparison Study of Surface Mesh Generation. *J. Mol. Model.* **2015**, *21*, No. 113.

(13) Weiser, J.; Shenkin, P.; Still, W. Optimization of Gaussian Surface Calculations and Extension to Solvent-Accessible Surface Areas. *J. Comput. Chem.* **1999**, *20*, 688−703.

(14) Wang, J.; Olsson, S.; Wehmeyer, C.; Perez, A.; Charron, N. E.; de Fabritiis, G.; Noe, F.; Clementi, C. Machine Learning of Coarse-Grained Molecular Dynamics Force Fields. *ACS Cent. Sci.* **2019**, *5*, 755−767.

(15) Yu, Z.; Jacobson, M. P.; Friesner, R. A. What Role Do Surfaces Play in GB Models? A New-Generation of Surface-Generalized Born Model Based on a Novel Gaussian Surface for Biomolecules. *J. Comput. Chem.* **2006**, *27*, 72−89.

(16) Lu, B. Z.; Zhou, Y. C.; Holst, M. J.; McCammon, J. A. Recent Progress in Numerical Methods for the Poisson-Boltzmann Equation in Biophysical Applications. *Commun. Comput. Phys.* **2008**, *3*, 973−1009.

(17) Chen, M.; Lu, B. Advances in Biomolecular Surface Meshing and Its Applications to Mathematical Modeling. *Chin. Sci. Bull.* **2013**, *58*, 1843−1849.
(18) Connolly, M. Analytical Molecular-Surface Calculation. *J. Appl. Crystallogr.* **1983**, *16*, 548−558.
(19) Lee, B.; Richards, F. M. The Interpretation of Protein Structures: Estimation of Static Accessibility. *J. Mol. Biol.* **1971**, *55*, 379−400.
(20) Richards, F. Areas, Volumes, Packing, And Protein-Structure. *Annu. Rev. Biophys. Bioeng.* **1977**, *6*, 151−176.
(21) Zhang, Y.; Xu, G.; Bajaj, C. Quality Meshing of Implicit Solvation Models of Biomolecular Structures. *Comput. Aided Geom. Des.* **2006**, *23*, 510−530.
(22) Liao, T.; Xu, G.; Zhang, Y. Atom Simplification and Quality T-mesh Generation for Multi-resolution Biomolecular Surfaces. In *Isogeometric Analysis and Applications*; Jüttler, B.; Simeon, B., Eds.; Lecture Notes in Computational Science and Engineering, Vol. *107*; Springer: Cham, Switzerland, 2014; pp 157−182.
(23) Carr, J. C.; Beatson, R. K.; Cherrie, J. B.; Mitchell, T. J.; Fright, W. R.; McCallum, B. C.; Evans, T. R. In *Reconstruction and Representation of 3D Objects with Radial Basis Functions*. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01, 2001.
(24) Samozino, M.; Alexa, M.; Alliez, P.; Yvinec, M. In *Reconstruction with Voronoi Centered Radial Basis Functions*. Proceeding of the Symposium on Geometry Processing, 2006.
(25) Li, M.; Chen, F.; Wang, W.; Tu, C. Sparse RBF Surface Representations. *Comput. Aided Geom. Des.* **2016**, *48*, 49−59.
(26) Elad, M. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed.; Springer Publishing Company, 2010.
(27) Määttä, J.; Schmidt, D. F.; Roos, T. Subset Selection in Linear Regression using Sequentially Normalized Least Squares: Asymptotic Theory. *Scand. J. Stat.* **2016**, *43*, 382−395.
(28) Taylor, H.; Banks, S.; Mccoy, J. Deconvolution with L1 Norm. *Geophysics* **1979**, *44*, 39−52.
(29) Rissanen, J. Modeling by Shortest Data Description. *Automatica* **1978**, *14*, 465−471.
(30) Grote, M.; Huckle, T. Parallel Preconditioning with Sparse Approximate Inverses. *SIAM J. Sci. Comput.* **1997**, *18*, 838−853.
(31) Girosi, F. An Equivalence between Sparse Approximation and Support Vector Machines. *Neural Comput.* **1998**, *10*, 1455−1480.
(32) Chen, S.; Donoho, D.; Saunders, M. Atomic Decomposition by Basis Pursuit. *SIAM J. Sci. Comput.* **1998**, *20*, 33−61.
(33) Daubechies, I.; Defrise, M.; De Mol, C. An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint. *Commun. Pure Appl. Math.* **2004**, *57*, 1413−1457.
(34) Xu, L.; Wang, R.; Zhang, J.; Yang, Z.; Deng, J.; Chen, F.; Liu, L. Survey on Sparsity in Geometric Modeling and Processing. *Graphical Models* **2015**, *82*, 160−180.
(35) Blinn, J. F. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graphics* **1982**, *1*, 235−256.
(36) Zhang, Y.; Xu, G.; Bajaj, C. Quality Meshing of Implicit Solvation Models of Biomolecular Structures. *Comput. Aided Geom. Des.* **2006**, *23*, 510−530.
(37) Weiser, J.; Shenkin, P. S.; Still, W. C. Optimization of Gaussian Surface Calculations and Extension to Solvent-Accessible Surface Areas. *J. Comput. Chem.* **1999**, *20*, 688−703.
(38) Bai, S.; Lu, B. VCMM: A Visual Tool for Continuum Molecular Modeling. *J. Mol. Graphics Modell.* **2014**, *50*, 44−49.
(39) Chen, M.; Tu, B.; Lu, B. Triangulated Manifold Meshing Method Preserving Molecular Surface Topology. *J. Mol. Graphics Modell.* **2012**, *38*, 411−418.
(40) Chen, M.; Lu, B. TMSmesh: A Robust Method for Molecular Surface Mesh Generation Using a Trace Technique. *J. Chem. Theory Comput.* **2011**, *7*, 203−212.
(41) Liu, T.; Chen, M.; Lu, B. Efficient and Qualified Mesh Generation for Gaussian Molecular Surface Using Adaptive Partition and Piecewise Polynomial Approximation. *SIAM J. Sci. Comput.* **2018**, *40*, B507−B527.
(42) Dolinsky, T.; Nielsen, J.; McCammon, J.; Baker, N. PDB2PQR: An Automated Pipeline for the Setup of Poisson-Boltzmann Electrostatics Calculations. *Nucleic Acids Res.* **2004**, *32*, W665−W667.
(43) Kingma, D. P.; Ba, J. A Method for Stochastic Optimization, arXiv:1412.6980. arXiv.org e-Print archive. https://arxiv.org/abs/1412.6980 (submitted Dec 22, 2014).
(44) PyTorch. https://pytorch.org.
(45) Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* **2010**, *3*, 1−122.
(46) Cignoni, P.; Rocchini, C.; Scopigno, R. Metro: Measuring Error on Simplified Surfaces. *Comput. Graphics Forum* **1998**, *17*, 167−174.
(47) Zhang, B.; Peng, B.; Huang, J.; Pitsianis, N. P.; Sun, X.; Lu, B. Parallel AFMPB Solver with Automatic Surface Meshing for Calculation of Molecular Solvation Free Energy. *Comput. Phys. Commun.* **2015**, *190*, 173−181.

https://dx.doi.org/10.1021/acs.jcim.0c00585
J. Chem. Inf. Model. 2020, 60, 6054−6064