10-31-2019

DEEPCON-PRE: Improved protein contact map prediction using inverse covariance and deep residual networks

Nachammai Palaniappan
University of Missouri-St. Louis, npnb7@mail.umsl.edu

DEEPCON-PRE: Improved protein contact map prediction using inverse covariance and deep residual networks

Nachammai Palaniappan

A THESIS
Presented to the Faculty of
The Graduate School at the University of Missouri – St. Louis
In partial Fulfillment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Dr. Badri Adhikari

St. Louis, MO

Abstract

As with most domains where machine learning methods are applied, correct feature engineering is critical when developing deep learning algorithms for solving the protein folding problem. Unlike the domains such as computer vision and natural language processing, feature engineering is not rigorously studied towards solving the protein folding problem. A recent research has highlighted that input features known as precision matrix are most informative for predicting inter-residue contact map, the key for building three-dimensional models. In this work, we study the significance of the precision matrix feature when very deep residual networks are trained. Using a standard dataset of 3456 proteins, known as the DeepCov set, we trained multiple deep residual networks and tested our models on an independent test dataset of 150 proteins. On this test dataset, we find that precision matrix features deliver 3.7% more precise long-range contacts than the benchmark covariance matrix features in our recently published method DEEPCON. In addition to validating the findings that precision matrix is more informative, we also find that the significance of precision matrix is reduced when deeper residual network models are trained. Our method, DEEPCON-PRE, i.e. DEEPCON with precision matrix as input feature, is available at https://github.com/nachammai779/Deepcon_Precision.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning is a subfield of machine learning. It is a mathematical framework to learn new representations of data. New increasingly meaningful representations from data are learned incrementally in every new successive layer. These layered representations are learned via models called neural networks. Neural networks transform the input data into representations that are increasingly different from the original data and increasingly informative about the final result. The deep learning networks typically perform automatic feature extraction without human intervention unlike most traditional machine-learning algorithms. Prior to the boom of the deep learning practices, feature extraction step was needed to manually engineer good layers of data representations. But these manual feature engineering methods take time for the scientists to get their input data ready. This method did not prove to be very successful particularly for image data because it was difficult to manually extract all the relevant features needed for the accurate classification. When training using unlabeled data, each node in a layer of a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network's guesses and the probability distribution of the input data. In the process, these deep neural networks learn to recognize correlations between certain relevant features and the actual output. They draw connections between feature signals and what those features represent. The process of recognizing correlations between relevant features and true output makes deep neural network

algorithms more useful compared to prior practices. In this work, we predict protein inter-residue contacts using standard two-dimensional convolutional neural networks, commonly used for image classification problems.

1.1 What is protein folding?

Protein folding is the process by which a protein structure assumes its functional shape or conformation. All protein molecules are heterogeneous unbranched chains of amino acids. By coiling and folding into a specific three-dimensional shape they can perform their biological function. Protein structure prediction is the inference of the three-dimensional structure of a protein from its amino acid sequence—that is, the prediction of its folding and its secondary and tertiary structure from its primary structure.

Scientists from multiple fields including machine learning, biology, physics, chemistry, and mathematics have combined their ideas and have come up with innovative solutions for protein structure prediction. One such idea is using the multiple sequence alignment files as input and predicting the residues in the proteins using different kinds of predictors. A multiple sequence alignment is a sequence alignment of three or more biological sequences, generally protein, DNA, or RNA. Multiple sequence alignments can be helpful in many circumstances like detecting historical and familial relations between sequences of proteins or amino acids and determining certain structures or locations on sequences. An amino acid is made up of a few different parts, connected to each other. The parts of an amino acid are an amine group, a carboxylic acid group, and the residue. The amine and

carboxylic acid groups give the name ‘amino acid,’ and these two parts are identical to those of other amino acids. The residue is the part that is unique among each of the 20 amino acids. Think of the generic definition of residue as something leftover. An amino acid residue is what's left over when you take away all the identical parts of the amino acid. Amino acid residues are important because they are the unique portion of an amino acid. They are the part that gives the amino acid its unique identity. When amino acids are lined up to form a protein, they’ll arrange themselves so that hydrophilic residues are exposed to water, and hydrophobic residues are hidden from water. This can cause the protein to form into an alpha-helix, which is a coiled-up shape, or to form into a beta pleated sheet, which is a zig-zag shape. The shape a protein takes is incredibly important for its function. Amino acid residues can also link pieces of a protein together. This is another way protein get its specific shape.

1.2 Inter-residue contact-map prediction

A protein contact map represents the distance between all possible amino acid residue pairs of a three-dimensional protein structure using a binary two-dimensional matrix. For two residues if the element of the matrix is 1 then the two residues are closer than a predetermined threshold or 0 otherwise. If they are close to each other in the true output, then the predicted residues must reflect the same. Typically, residues are defined to be in contact when the distance between their β-carbon atoms (or α-carbon for the amino acid glycine) is smaller than 8 Å (Moult, Fidelis, Kryshtafovych, Schwede, & Tramontano, 2018) (Monastyrskyy, D’Andrea, Fidelis, Tramontano, & Kryshtafovych, 2016). The

ability to make these predictions about the residues can assist researchers by providing information about the native structure and other physical properties of that protein.

Protein contact prediction problem is gaining momentum in the scientific community. The Critical Assessment of protein Structure Prediction (CASP) experiments aim at establishing the current state of the art in protein structure prediction. Their goal is to help advance the methods of identifying protein structure from sequence. They conduct biannual protein structure prediction competitions. In the most recent CASP13 experiment (Moult et al., 2018) DeepMind's Alphafold topped in predicting the protein structures. The key idea of AlphaFold's approach is that a distribution over pairwise distances between residues corresponds to a potential that can be minimized using gradient descent after being turned in to a continuous function (Jinbo Xu & Wang, 2019). Before the widespread use of machine learning methods, contacts were predicted from protein sequence alignments based on the principle that evolutionary pressures place constraints on the sequence evolution over generations(Marks et al., 2011). In recent past the most successful methods are those that combine the predicted features and the convolutional neural network model. Convolutional neural networks have proved to be of vital importance because of their inherent nature to perform cross-correlation operations to learn more features. Similarly, DNCON2 (Adhikari, Hou, & Cheng, 2018) uses two-level deep convolutional neural network and arrives at better prediction of protein contacts. It consists of six convolutional neural networks-the first five predict contacts at 6, 7.5, 8, 8.5 and 10 Å distance thresholds, and the last one uses these five predictions as additional features to predict final contact maps. PSICOV (Jones, Buchan, Cozzetto, & Pontil, 2012) uses deeper and wider first-

stage network architecture composed of two hidden layers of 160 ReLU units and make contact map predictions.

1.3 Features for protein contact prediction

In the past decade many attempts to feature engineer the predicted multiple sequence alignments using different kinds of matrices have been successful. Pair-frequency matrix and covariance matrix computed from the sequence alignments are two examples. A covariance matrix is one where the off-diagonal elements contain the covariances of each pair of variables. The diagonal elements of the covariance matrix contain the variances of each variable. The variance measures how much the data is scattered about the mean. Thus, the covariance matrix can only capture marginal correlations among variables. An example of a contact prediction method that uses covariance matrix is DEEPCON (Adhikari, 2019). It uses covariance matrix and residual neural network architecture with dilation and dropout (RDD) to make contact map predictions. Residual neural networks use short cuts also known as skip connections to allow very deep neural networks to learn efficiently without hurting the performance of the model. This is achieved by taking the activation from one layer and feeding it into another much deeper layer in the neural network. This unit is called the residual block. DEEPCON uses dilated convolution in order to capture global view of the input. This increases the receptivity of the network and captures more contextual information (Onvolutions, 2016). Dropout is a technique to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. This helps to reduce overfitting of the models that is common

when we train very deep neural networks that typically have large number of parameters (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

In a recent study, authors demonstrate that precision matrix are more informative than covariance matrix (Li, Hu, Zhang, Yu, & Zhang, 2019). This ResPRE method uses precision matrix and ResNet architecture to predict contact maps. ResNet is a deep neural network architecture introduced by Microsoft in their winning entry in the ILSVRC ImageNet challenge (He, Zhang, Ren, & Sun, 2016). The inverse covariance matrix, or the precision matrix, displays information about the partial correlations of variables. With the covariance matrix one observes the unconditional correlation between a variable i, to a variable j by reading the $(i,j)^{th}$ index. It may be the case that the two variables are correlated, but do not directly depend on each other and another variable k explains their correlation. But if one will condition on the variable k, then the two variables i and j become partially correlated. A partial correlation describes the correlation between variable i and j once you condition on all other variables. If i and j are conditionally independent then the $(i,j)^{th}$ element of the precision matrix will equal zero. So, the inverse covariance matrix (or precision matrix) is best used to describe the conditional independent relationships among all variables.

1.4 Contact prediction is computationally intensive

Conceptually, the protein contact prediction problem is like the depth prediction problem in computer vision. In the depth prediction problem, the input is an image of dimensions H x W x C, where H is height, W is width, and C is number of input channels, and output is a two dimensional matrix of size H x W whose values represent the depth intensities(Eigen, Puhrsch, & Fergus, 2014). Similarly, in the protein contact prediction the length of a protein in a multiple sequence alignment file is L and there are N input channels. Hence the input is protein features of dimension L x L x N and the output is a contact probability map (matrix) of size L x L. Computer vision problems have three channels (red, green, and blue or hue, saturation, and value) while the latter, contact prediction problem, has much higher number of channels like 441 (Jones & Kandathil, 2018). For a given pair of amino acid types, pair frequencies or covariances are composed as an $m \times m$ matrix, where m is the number of columns in the sequence alignment. Considering 20 amino acid types and possible gaps, there are $(20+1) \times (20+1) = 441$ such matrices for a given alignment. These 441 matrices are presented (in image recognition terms) as feature channels in the input to the convolutional neural networks. As the number of channels increase, the input volume becomes large and this leads to longer training times. Another unique feature of the contact prediction problem is that the protein sequences are of varying length, unlike fixed size input images in computer vision problems. Consequently, the input features are of varying sizes. Since every bit (amino acid) of the raw MSA file is fixed at that position, there is no possibility to rearrange or

alter or augment their sequence. This further complicates the training process where a deep learning model expects a fixed size input volume.

In this work, we aim to investigate the significance of the precision matrix features compared to the covariance matrix feature and validate the findings of the ResPRE methods using a different DeepCov dataset, particularly when very deep models such as DEEPCON is trained. We calculate our precision matrix using a ridge predictor (Li et al., 2019). Ridge regression (Hoerl, A.E. and Kennard, R. (1970)) uses a type of shrinkage estimator called a ridge estimator. Shrinkage estimators theoretically produce new estimators that are shrunk closer to the "true" population parameters. The ridge estimator is especially good at improving the least-squares estimate when multicollinearity is present. Multicollinearity exists whenever an independent variable is highly correlated with one or more of the other independent variables in a regression equation. Multicollinearity is a problem because it undermines the statistical significance of an independent variable. Ridge regression is a linear regression model whose coefficients are estimated by the ridge estimator, is biased and has lower variance than the least squares estimator. The lower variance of the ridge estimators results in better predictive models. In order to validate our results, we repeat our contact prediction training and testing experiments for the two input features (covariance and precision matrix) each derived using two implementations (DEEPCON and ResPRE).

Chapter 2

Materials and Methods

2.1 Datasets

In our experiments we used publicly available DeepCov dataset consisting of 3456 proteins (Jones & Kandathil, 2017) publicly available at https://github.com/psipred/DeepCov. Following the recent practice of setting aside validation sets from the training examples, we use the first 130 proteins, when the PDB IDs are sorted alphabetically, in the dataset as our validation set. The remaining set of proteins were used for training. We trained and validated our models using the DeepCov dataset and tested the models on an independent dataset known as PSICOV dataset consisting of 150 representative proteins (Jones et al., 2012). **Figure 2.1** summarizes our experimental setup.

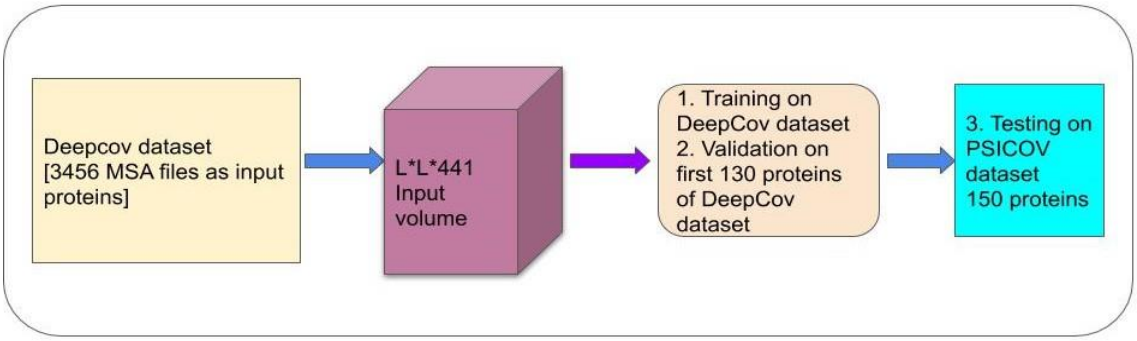

Figure 2.1: In our experiments, we train and validate using the DeepCov dataset, where each input protein is an L*L*441 matrix, and test using the PSICOV 150 proteins.

2.2 Contact evaluation

We followed the definition and categorization of contact predictions as per the conventional criterions in CASP experiments (Moult et al., 2018). A residue pair whose Euclidean distance between two C-beta (Ca for Glycine) atoms is smaller than 8 Angstroms is considered as a contact. Residue pairs in contact and separated by at least 24 residues in the sequence are long-range contacts, whereas those with a sequence separation between 12 and 23 or 6 and 11 are considered as medium- or short-range contacts, respectively. In this work, we evaluate the precision of top L/5, L/2 and L (the length of the protein sequence) for three different types of contacts (short-, medium- and long range) as the major evaluation metrics. This same evaluation metric is used on other standard methods such as MetaPSICOV, ResPRE, and DEEPCON. Occasionally, we also evaluate the precision of all medium- and long-range contacts. We calculate the precision of top X contacts (P_X) using the following technique. We first rank all the predicted contacts in a contact map by the predicted probability score and select X number of top contact pairs. Precision is then the percentage of the correct predictions among these X predicted contact pairs. Similarly, for evaluating the precision of all medium- and long-range contacts ($P_{ALL-MLR}$) for a protein with total N_{LR} number of medium- and long-range contacts, we first round the top N_{LR} medium- and long-range predicted probabilities (after ranking the probabilities). Precision is then calculated as the ratio of 'the number of matches between predicted and true matrix' and $N_{LR.}$

2.3 Three-dimensional model evaluation

We use TM-score (Jinrui Xu & Zhang, 2010) to evaluate predicted models. TM-score is a metric for measuring the similarity of two protein structures. It is designed to solve two major problems in traditional metrics such as root-mean-square deviation (RMSD): (1) TM-score measures the global fold similarity and is less sensitive to the local structural variations; (2) magnitude of TM-score for random structure pairs is length-independent. TM-score has the value in (0,1), where 1 indicates a perfect match between two structures. Following strict statistics of structures in the PDB (protein data bank), scores below 0.17 correspond to randomly chosen unrelated proteins whereas structures with a score higher than 0.5 assume generally the same fold.

2.4 Input features

We generate covariance matrix and precision matrix using two techniques – extension of the DEEPCON implementation and extension of the ResPRE implementation. The first method of generating the covariance and precision matrix involves extending the Python version of the ‘cov21stats’ program in the DeepCov package (Jones & Kandathil, 2017) available in the DEEPCON package. From the input multiple sequence alignment, at first the probabilities of observing every pair of the 20 amino acids are calculated, with gap characters considered as an additional amino acid category. Frequencies for unobserved residue pairs are estimated with a pseudo count of 1. Sequence clusters are weighted based on a 62% sequence identity clustering threshold. Using the marginal and pair frequencies

for each pair of amino acids as described above, the covariance between every pair of residues at every pair of sites is calculated. For a given pair of amino acid types, covariance matrix will be of the shape n*n, where n is the number of columns in the sequence alignment file. Considering that 20 amino acid types plus a category for gap, there are 21*21=441 such matrices for a given sequence alignment file. These 441 matrices are presented as feature channels as the input to the convolutional neural networks.

Calculating a precision matrix involves taking the inverse of the covariance calculations. Since matrices could be singular because the determinant values tend to be zero, this results in non-invertible matrices. When this is the case there could theoretically be infinite correct solutions, in other words there isn't a closed form solution, hence we need to use an estimator such as ridge regularized estimator to solve this problem. Linear estimators like least squares is too sophisticated for the input data that we have at hand. This is because our input data is non-linear and multi-dimensional by nature. The task of regularization is to constrain the learning to prevent overfitting the data. By constraining the learning algorithm to select 'simpler' hypotheses (prediction) from a possible set of hypotheses, we sacrifice a little bias for a significant gain in the variance. We estimate the precision matrix through the maximum likelihood approach (Friedman, Hastie, & Tibshirani, 2008)(Kuismin, Kemppainen, & Sillanpää, 2017)(Van Wieringen & Peeters, 2016). Estimates obtained from classical machine learning approaches will be accurate based on the important principle that the input variables come from independently and identically distributed data. Based on this assumption the precision matrix (θ) is calculated by minimizing the regularized log-likelihood function of

$$G = \text{tr}\,(S\theta) - log|\theta| + R(\theta)$$

The first two terms are the negative log likelihood of θ under the assumption that the data follows the gaussian multivariate distribution, tr(Sθ) is the trace of matrix Sθ, where S is the covariance matrix; $\log|\theta|$ is the log determinant of θ; R(θ) is the regularization function over θ to avoid overfitting. Regularization parameter used is e^{-6}. G is the convex function we want to minimize. To do this the derivative of G is taken and equated to zero and we derive an equation for the covariance matrix in terms of precision matrix.

$$S - \theta^{-1} + 2\rho\,\theta = 0$$

The covariance matrix S has the same eigen vectors and eigen values as $\theta^{-1} + 2\rho\,\theta$. Eigen decomposition is performed on both sides of the above equation to arrive at a ridge regularized solution. This method is adapted based on ResPRE (Li et al., 2019). Precision matrix features are richer features because they describe the conditional independent relationships among all variables(Li et al., 2019). 441 precision features (channels) were thus calculated. As our second method for generating covariance matrix and precision matrix, we extended the ResPRE's implementation at https://github.com/leeyang/ResPRE. For obtaining the dimensionality of the covariance matrix from this implementation, we implemented an appropriate conversion of our LxLx441 dimension matrix to a matrix of dimensions Lx21 by Lx21.

2.5 Residual Neural network architecture

As our deep neural network we used the existing implementation of the DEEPCON method (Adhikari, 2019). This convolutional neural network architecture has an input layer, many 2D convolutional layers with batch normalization or dropouts, residual connections and rectified linear units (ReLU) activations. The final layer is a convolutional layer with one filter of size 3x3 followed by a ‘sigmoid’ activation to predict contact probabilities.

Figure 2.2 visualizes our network architecture. To increase the stability of a neural network, batch normalization layers normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. In order to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. This is unlike the ‘tanh’ and ‘sigmoid’ activation functions that learn to approximate a zero output, e.g. a value very close to zero, but not a true zero value. This means that negative inputs can output true zero values allowing the activation of hidden layers in neural networks to contain one or more true zero values. This is called a sparse representation and is a desirable property in representational learning as it can accelerate learning and simplify the model.

The rectified linear activation function is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The intermediate layers will use ReLU as their activation function, and the final layer will use a sigmoid activation so as to output a probability (a score between 0 and 1,indicating how likely the sample is to

have the target “1”: how likely the review is to be positive). A rectified linear unit is a function meant to zero out negative values, whereas a sigmoid “squashes” arbitrary values into the [0, 1] interval, outputting something that can be interpreted as a probability.

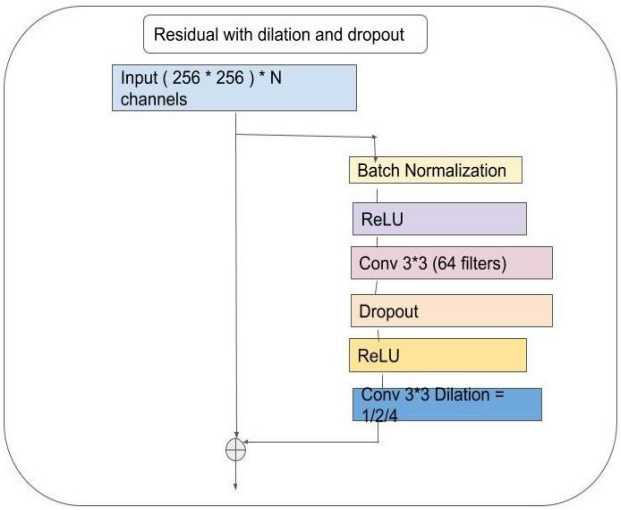

Figure 2.2 The DEEPCON residual network architecture used in our experiments. The last layer in the block is a dilated convolutional layer with dilation rate of 1, 2 and 4 at alternating blocks.

In the convolutional layers of our architecture, the variables are the numbers and size of filters at each layer, and dilation rate when dilated convolutional layers are used. All the CNN filters in the first layer convolve through the input volume of 256 x 256 x N producing batch normalized and ReLU activated outputs passed as input to the subsequent layers. The number of channels N is 441. We stop training the model if the validation accuracy does not improve for 20 epochs and reduce the learning rate by 0.5 when the loss does not improve for 10 epochs. Error is computed using binary cross entropy calculated as $-(y \log(p)+(1-y) \log(1-p))$, where p is the output of the sigmoid activation of the last layer for each residue pair, and y is 1 if the residue pair are in contact in the experimental structure

or else is 0. There are different loss functions in deep learning models. But crossentropy is usually the best choice when we are working with models that output probabilities.

Binary cross entropy is used as the loss function. Crossentropy is a quantity from the field of information theory that measures the distance between probability distributions or, in this case, between the ground-truth distribution and the actual predictions. Although the input length of our training proteins to 256, after training the model can make predictions for a protein of any length. Since contact matrix is symmetrical, we average the prediction of either triangle to generate final predictions. The residual block consisting of two convolutional layers, each preceded by a batch normalization layer and ReLU activation. Fixing the total number of convolutional filters in each layer to 64, the second batch normalization layer is replaced with a dropout layer and the second convolutional layer is replaced with a dilated convolution layer at alternating dilation rates of 1, 2, and 4.

2.6 Experiments

To study the significance of precision matrix over covariance matrix we trained the DEEPCON model using precision matrix and covariance matrix obtained using our extension of the DEEPCON method and the ResPRE method. To obtain maximum precision for each of these four experiments we repeated some experiments by varying the number of residual blocks in our architecture. We used keras library (https://keras.io/) with TensorFlow (https://www.tensorflow.org/) backend for our training and testing. On a NVIDIA Quadro P6000 GPU with 24 GB GPU memory, one training experiment (32-45

epochs) took about up to 24 hours. This training time could only be achieved with the use of Solid-State Drives (SSDs).

Chapter 3

Results

3.1 Precision matrix is more predictive

As our first experiment, we investigated the performance gain in predicting protein inter-residue contacts by replacing the covariance matrix with the precision matrix in the original DEEPCON method. To evaluate our DEEPCON-PRE method, we compare its performance against its own predecessor DEEPCON method that uses covariance matrix as input. Our results (see **Figure 3.1**) indicate the performance gain obtained while using the precision matrix calculations in place of covariance methods on the PSICOV150 test dataset. Few samples indicate extremely high prediction accuracy when using DEEPCON-PRE. Best samples show 33.3%, 17.4% 15.6% improvement in prediction accuracy. The average improvement in prediction accuracy is 3.69% in DEEPCON-PRE as compared to DEEPCON when evaluation top L/5 long-range contacts.

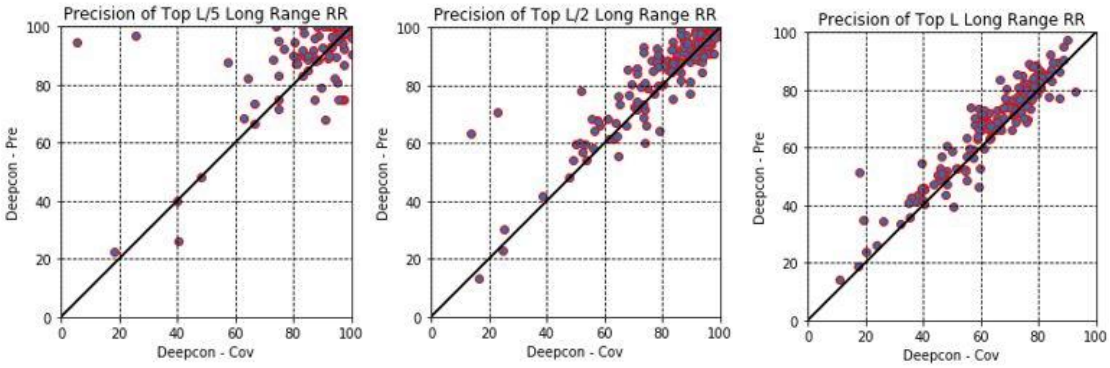

Figure 3.1: Comparison of the performance of our DEEPCON-PRE method with the original DEEPCON method using the precision of top L/5, top L/2, and top L contacts as the evaluation metric. Scatter plot with more data points in the upper triangle demonstrate that our method performs better.

To substantiate our findings, we repeated our experiments by generating covariance matrix and precision matrix calculations using the ResPRE implementation and train the same DEEPCON model. Our results on the independent test dataset of 150 proteins, the PSICOV set, show that the precision matrix delivers at least 3% more precise contacts when compared to covariance on all the three metrics – $P_{L/5}$, $P_{L/2}$ and P_L (see **Table 1**). This clearly indicates that the precision matrix-based features outperform the covariance features. These results are also observed on the validation datasets (see Tables in the Appendix).

Table 1 Mean precision of top L/5, L/2, and L long-range contacts ($P_{L/5}$, $P_{L/2}$, and P_L, respectively) on the 150 protein in the PSICOV test dataset when covariance matrix and precision matrix features using the DEEPCON and ResPRE implementations are used as input for training.

Implementation	Input Feature	$P_{L/5}$	$P_{L/2}$	P_L
DEEPCON	Covariance Matrix	90.9	79.7	63.2
DEEPCON	Precision Matrix	**93.3**	**83.3**	**66.9**
ResPRE	Covariance Matrix	89.3	77.3	60.0
ResPRE	Precision Matrix	**93.8**	**84.6**	**69.1**

3.2 Learning curves

Next, we study how the change in the number of blocks in our network, i.e. network depth, affects the precision of precision matrix. For this, we trained our model using varying number of residual blocks such as 4, 8, 16, etc. up to 40. The maximum depth is set to 40 as our GPU did not consistently support the training beyond 42 residual blocks. **Figure 3.2** shows how the precision of top L/5, L/2 and L long-range contacts improve over the training epochs. Consistent with general understanding in machine learning that training for more epochs yields better performance, we observe that the performance of both – covariance matrix and precision matrix – consistently increase for the first few epochs and decelerate afterwards. It is worth noting that the precision-matrix based model consistently outperforms the covariance-matrix based model. To verify our findings, we repeated the same experiments with ResPRE implementation of generating covariance matrix and precision matrix and observed the same trend. This extensive study leads us to conclude that precision matrix is consistently more informative than covariance matrix across all network depths and evaluation metrics.

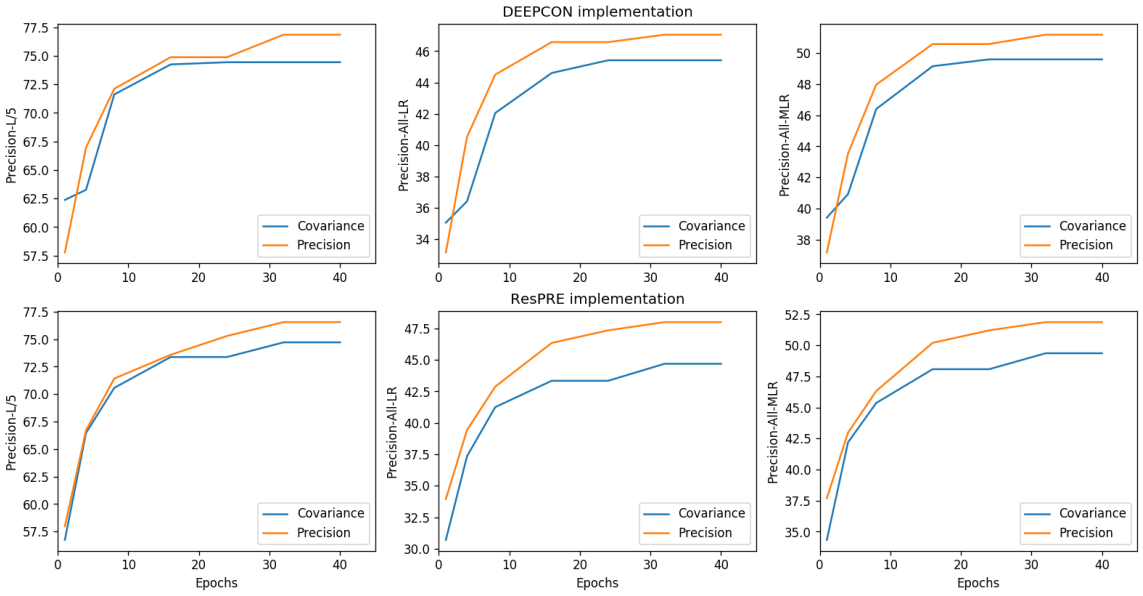

Figure 3.2: Precision of top L/5 and top L long-range contacts (first two columns) and all medium- and long-range contacts (last column) over the training epochs when covariance matrix and precision matrix are used as input feature using the DEEPCON implementation (top row) and ResPRE implementation (bottom row). Results are shown on the validation dataset.

3.3 Improved contacts yield more accurate models

For some selected proteins in the test dataset, we predicted contacts using the two methods - DEEPCON-PRE and DEEPCON and built three-dimensional protein models using the CONFOLD2 method (Adhikari & Cheng, 2018). It builds models using various subsets of input contacts to explore the fold space under the guidance of a soft square energy function, and then clusters the models to obtain the top five models. We visually compared the predicted models and evaluated them using TM-score. As an example, we pick the chain A of the protein '1k7j' compare the model built using DEEPCON contacts and DEEPCON-

PRE contacts(**Figure 3.3**). Here we choose an example where DEEPCON-PRE generates more accurate contacts in order to check if improved contacts lead to improved models. We find that the TM-score of the top models generated by DEEPCON and DEEPCON-PRE are 0.19 and 0.29 respectively. For this protein the contact precision of top L long-range contacts (P_L) by DEEPCON and DEEPCON-PRE are 56.3 and 73.8.

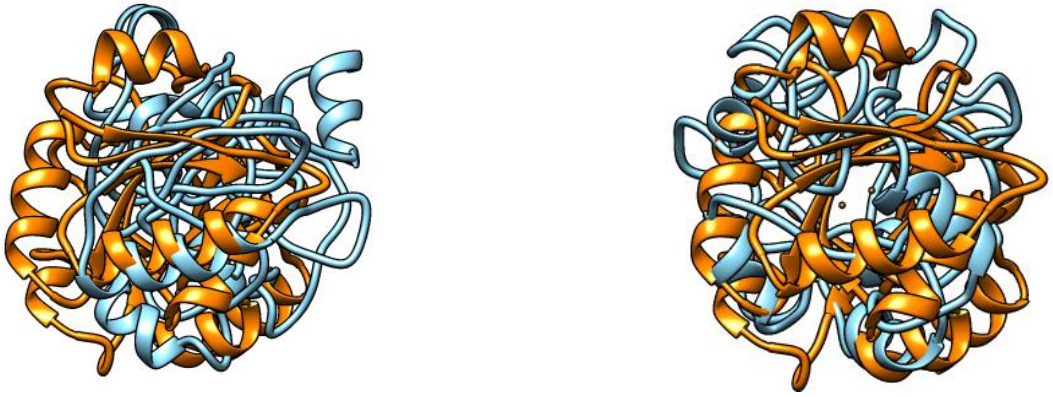

Figure 3.3: Top one models built using DEEPCON (blue-left) and DEEPCON-PRE (blue-right) contacts for the protein '1k7j' superposed on the native structure (orange). TM-score/RMSD of the DEEPCON model and DEEPCON-PRE models are 0.19/17.3 and 0.29/5.9 respectively.

Similarly, to check if the models built using the precision matrix obtained using the ResPRE implementation were better than the ones built using covariance matrix, we picked the chain A of the protein '1jbk' from the test dataset(**Figure 3.4**). We find that the TM-score of the top models generated by ResPRE-COV and ResPRE-PRE are 0.2047 and 0.2017 respectively. For this protein the contact precision of top L long-range contacts (P_L) by ResPRE-COV and ResPRE-PRE are 30.69 and 68.78.

Figure 3.4: Top one models built using ResPRE-COV (blue-left) and ResPRE-PRE (blue-right) contacts for the protein '1jbk' superposed on the native structure (orange). TM-score/RMSD of the ResPRE-COV model and ResPRE-PRE models are 0.2047/17.0 and 0.2017/16.4 respectively.

3.4 Deeper networks do not necessarily perform better

Contrary to the findings by the authors of the ResPRE method, we find that deeper models do not necessarily favor the precision matrix features. While deeper networks generally performed better with covariance matrix as input, the models saturated at smaller depths when precision matrix is used as input. The overall trend we observed (see **Figure 3.5**) shows that the performance using precision matrix saturates at a depth less than the maximum depth we tested. This leads to many questions such as why a deeper residual network with more parameters performs more poorly than a shallower network with fewer parameters. Although such an investigation is out of scope for this work, we speculate that the input feature value distribution of a precision matrix is the cause. We believe that better ways to normalize the input precision matrix may resolve this issue allowing deeper models to perform better.

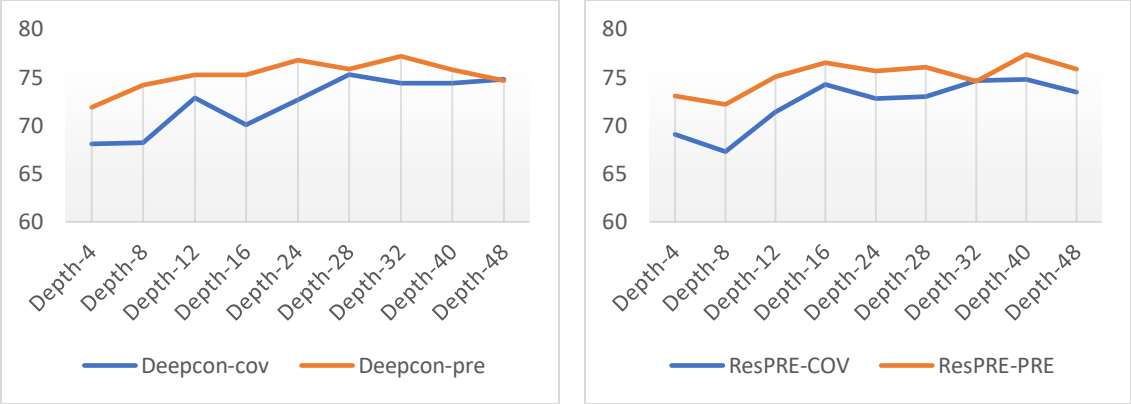

Figure 3.5: Plots showing the precision of top L/5 long-range contacts vs the network depth (number of residual blocks) when covariance matrix and precision matrix are used as input feature in the DEEPCON implementation (left) and ResPRE implementation (right).

Chapter 4

Conclusions and future work

We found that the features generated by the precision matrix performed better consistently than the covariance matrix features in our deep residual network architecture. Improved predictions were observed across all-range contacts. This increased performance of precision features was observed even when we used ResPRE-style methods for validating our findings. Our DEEPCON-PRE gave 3.7% improvement in prediction accuracy when compared to the original DEEPCON method. Similarly, our model showed a 9.04% improvement in prediction accuracy when the ResPRE implementation was used. This improved contact-map prediction also leads to improved three-dimensional models. We also find that our model's prediction accuracy did not improve significantly in deeper neural networks. As a future work, we believe that improved batch normalization techniques may resolve the issue of performance saturation while training deep neural networks. This could be achieved by alternative ways to normalize the layer inputs (Ioffe & Szegedy, 2015).

Acknowledgements

I would like to acknowledge Dr. Badri Adhikari PhD., for his valuable time and mentoring that he gave throughout my research work. I would like to acknowledge Dr. Uday Chakraborty PhD., for teaching me the core machine learning concepts that helped me develop insights and deeper understanding throughout my research work. I would like to acknowledge Dr. Sharlee Climer PhD., for her valuable suggestions during this work. I also would like to acknowledge Dr. Sanjiv Bhatia PhD., and Dr. Cezary Janikow PhD., for granting special permission to get this work done as part of summer semester. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU that were used for the entire computations during this research work.

Bibliography

Adhikari, B. (2019). DEEPCON: protein contact prediction using dilated convolutional neural networks with dropout. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btz593

Adhikari, B., & Cheng, J. (2018). CONFOLD2: Improved contact-driven ab initio protein structure modeling. *BMC Bioinformatics*. https://doi.org/10.1186/s12859-018-2032-6

Adhikari, B., Hou, J., & Cheng, J. (2018). DNCON2: Improved protein contact prediction using two-level deep convolutional neural networks. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btx781

Adhikari, B., Nowotny, J., Bhattacharya, D., Hou, J., & Cheng, J. (2016). ConEVA: A toolbox for comprehensive assessment of protein contacts. *BMC Bioinformatics*. https://doi.org/10.1186/s12859-016-1404-z

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*. https://doi.org/10.1093/biostatistics/kxm045

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.90

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*.

Jones, D. T., Buchan, D. W. A., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btr638

Jones, D. T., & Kandathil, S. M. (2017). DeepCov: Deep convolutional neural networks for protein contact prediction from simple alignment statistics alone Journal: *Bioinformatics*. https://doi.org/10.1093/bioinformatics/xxxxx

Jones, D. T., & Kandathil, S. M. (2018). High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics*.

https://doi.org/10.1093/bioinformatics/bty341

Jones, D. T., Singh, T., Kosciolek, T., & Tetchner, S. (2015). MetaPSICOV: Combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btu791

Kuismin, M. O., Kemppainen, J. T., & Sillanpää, M. J. (2017). Precision Matrix Estimation With ROPE. *Journal of Computational and Graphical Statistics*. https://doi.org/10.1080/10618600.2016.1278002

Li, Y., Hu, J., Zhang, C., Yu, D.-J., & Zhang, Y. (2019). ResPRE: high-accuracy protein contact prediction by coupling precision matrix with deep residual neural networks. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btz291

Marks, D. S., Colwell, L. J., Sheridan, R., Hopf, T. A., Pagnani, A., Zecchina, R., & Sander, C. (2011). Protein 3D structure computed from evolutionary sequence variation. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0028766

Monastyrskyy, B., D'Andrea, D., Fidelis, K., Tramontano, A., & Kryshtafovych, A. (2016). New encouraging developments in contact prediction: Assessment of the CASP11 results. *Proteins*. https://doi.org/10.1002/prot.24943

Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T., & Tramontano, A. (2018). Critical assessment of methods of protein structure prediction (CASP)—Round XII. *Proteins: Structure, Function and Bioinformatics*. https://doi.org/10.1002/prot.25415

Onvolutions, D. I. C. (2016). M -s c a d c. *Iclr*. https://doi.org/10.16373/j.cnki.ahr.150049

Schapire, R. E. (2003). *The Boosting Approach to Machine Learning: An Overview*. https://doi.org/10.1007/978-0-387-21579-2_9

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Van Wieringen, W. N., & Peeters, C. F. W. (2016). Ridge estimation of inverse covariance matrices from high-dimensional data. *Computational Statistics and Data Analysis*. https://doi.org/10.1016/j.csda.2016.05.012

Xu, Jinbo, & Wang, S. (2019). Analysis of distance-based protein structure prediction by deep learning in CASP13. *Proteins: Structure, Function, and Bioinformatics*. https://doi.org/10.1002/prot.25810

Xu, Jinrui, & Zhang, Y. (2010). How significant is a protein structure similarity with TM-score = 0.5? *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btq066

Appendix

Suppl. Table 1 Performance comparison between long-range, medium range and short-range contacts when using DEEPCON implementation for generating covariance and precision matrix were used on the validation dataset.

$\mathbf{P_{L/5}}$	Cov	74.4
	Pre	76.8
$\mathbf{P_{LR}}$	Cov	45.4
	Pre	47.0
$\mathbf{P_{MLR}}$	Cov	49.5
	Pre	51.1

Suppl. Table 2 Performance comparison between long-range, medium range and short-range contacts when using ResPRE implementation for generating covariance and precision matrix were used on the validation dataset.

$\mathbf{P_{L/5}}$	Cov	74.7
	Pre	76.5
$\mathbf{P_{LR}}$	Cov	44.6
	Pre	48.0
$\mathbf{P_{MLR}}$	Cov	49.3
	Pre	51.8

Suppl. Table 3 All-range contact-map predictions for 150 PSICOV test dataset in DEEPCON and DEEPCON-PRE evaluated using ConEVA.

	DEEPCON			DEEPCON-PRE		
PDB ID	**$P_{L/5}$**	**$P_{L/2}$**	**P_L**	**$P_{L/5}$**	**$P_{L/2}$**	**P_L**
1a3aA	93.1	89.04	73.1	96.55	91.78	80.69
1a6mA	100	89.47	72.19	100	90.79	74.17
1a70A	100	87.76	69.07	100	95.92	80.41
1aapA	90.91	71.43	48.21	100	85.71	60.71
1abaA	100	81.82	60.92	100	88.64	70.11
1ag6A	100	96	79.8	100	94	86.87
1aoeA	81.58	77.08	63.54	89.47	77.08	65.62
1atlA	97.5	74	59.5	75	60	46.5
1atzA	100	92.11	77.33	100	94.74	73.33
1avsA	87.5	56.1	32.1	75	58.54	33.33
1bdoA	100	97.5	90	100	100	97.5
1bebA	25.81	23.08	17.95	96.77	70.51	51.28
1behA	97.3	89.13	66.85	97.3	89.13	67.93
1bkrA	90.91	64.81	37.96	68.18	55.56	41.67
1brfA	100	92.59	60.38	100	88.89	66.04
1bsgA	96.23	89.47	68.8	98.11	90.98	76.69
1c44A	100	96.97	71.21	100	90.91	74.24
1c52A	100	90.32	66.67	100	91.94	68.29
1c9oA	73.08	57.58	38.17	88.46	68.18	43.51
1cc8A	100	94.44	84.72	100	100	83.33
1chdA	92.5	83.84	77.78	100	94.95	88.38
1cjwA	96.97	83.13	62.05	96.97	84.34	71.08
1ckeA	90.48	79.25	65.57	100	89.62	73.11
1ctfA	100	100	92.65	100	97.06	79.41
1cxyA	93.75	78.05	58.02	100	87.8	67.9
1cznA	94.12	87.06	79.29	100	97.65	85.21
1d0qA	75	64.71	47.06	95	76.47	49.02
1d1qA	96.88	90	79.25	96.88	93.75	88.68
1d4oA	97.14	76.4	59.32	100	88.76	73.45
1dbxA	83.33	73.68	66.45	100	93.42	83.55
1dixA	95.24	75	54.81	97.62	78.85	52.4
1dlwA	100	86.21	58.62	100	84.48	62.93
1dmgA	100	89.53	65.7	100	88.37	70.35
1dqgA	18.52	16.42	11.19	22.22	13.43	14.18
1dsxA	88.24	70.45	48.28	100	72.73	47.13
1eazA	95.24	63.46	45.63	80.95	61.54	48.54
1ej0A	100	90	71.11	97.22	90	76.67
1ej8A	64.29	55.71	46.43	82.14	68.57	57.14
1ek0A	97.06	91.67	76.79	100	98.81	89.29
1f6bA	88.57	81.82	65.91	88.57	85.23	68.75
1fcyA	80.85	58.47	40.25	87.23	66.1	45.76
1fk5A	63.16	53.19	39.78	68.42	59.57	41.94
1fl0A	57.58	50	39.63	87.88	59.76	45.12
1fnaA	100	100	73.63	100	97.83	75.82
1fqtA	95.45	78.18	61.47	95.45	92.73	71.56
1fvgA	97.37	93.75	74.48	86.84	84.38	78.12
1fvkA	86.84	73.4	55.32	92.11	79.79	63.3
1fx2A	100	85.71	55.36	100	87.5	57.14
1g2rA	84.21	74.47	47.87	89.47	65.96	43.62
1g9oA	100	91.3	74.73	100	97.83	82.42

1gbsA	91.89	64.52	37.3	89.19	62.37	41.08
1gmiA	100	97.06	86.67	100	98.53	89.63
1gmxA	100	85.19	71.03	100	90.74	71.03
1guuA	80	48	24	90	48	26
1gz2A	96.43	89.86	71.01	75	81.16	65.94
1gzcA	95.83	96.67	83.26	97.92	96.67	84.94
1h0pA	100	98.9	75.82	100	98.9	82.42
1h2eA	100	91.35	78.26	100	95.19	84.06
1h4xA	95.45	85.45	76.36	100	87.27	70
1h98A	66.67	56.41	40.26	73.33	64.1	45.45
1hdoA	100	95.15	83.9	100	98.06	86.34
1hfcA	100	78.48	57.32	100	79.75	56.69
1hh8A	5.26	13.54	19.27	94.74	63.54	34.9
1htwA	100	94.94	74.05	100	97.47	71.52
1hxnA	40.48	24.76	17.62	26.19	22.86	18.57
1i1jA	100	79.25	54.72	90.48	64.15	51.89
1i1nA	95.56	80.36	62.95	95.56	86.61	72.32
1i4jA	100	96.36	83.64	100	96.36	85.45
1i58A	93.1	69.44	57.64	100	80.56	59.03
1i5gA	86.84	65.26	56.08	89.47	73.68	59.79
1i71A	94.12	73.81	51.81	82.35	71.43	53.01
1ihzA	85.19	83.82	67.65	88.89	86.76	73.53
1iibA	100	94.23	79.61	100	92.31	78.64
1im5A	80.56	83.33	72.63	94.44	87.78	77.65
1iwdA	83.72	71.3	60	97.67	84.26	68.37
1j3aA	100	93.85	83.72	100	92.31	77.52
1jbeA	100	93.65	80.95	100	93.65	80.95
1jbkA	94.74	68.42	46.03	89.47	76.84	52.91
1jfuA	88.57	86.36	71.59	97.14	81.82	72.73
1jfxA	93.02	86.24	71.43	93.02	88.07	73.27
1jkxA	100	97.14	79.9	100	94.29	80.38
1jl1A	100	89.47	80.92	100	92.11	86.18
1jo0A	100	85.71	67.01	100	85.71	65.98
1jo8A	83.33	72.41	50	91.67	79.31	58.62
1josA	100	94	74	100	88	72
1jvwA	100	98.75	81.25	100	98.75	84.38
1jwqA	100	96.67	88.83	100	97.78	94.97
1jyhA	96.77	97.44	84.52	100	97.44	87.1
1k6kA	100	87.32	68.31	96.43	95.77	73.94
1k7cA	97.87	89.74	76.39	97.87	91.45	75.11
1k7jA	90.24	67.96	56.31	92.68	85.44	73.79
1kidA	97.44	89.69	75.65	92.31	77.32	68.91
1kq6A	75	51.43	35.71	71.43	60	42.86
1kqrA	81.25	73.75	50.62	87.5	68.75	39.38
1ktgA	74.07	78.26	68.61	100	88.41	77.37
1ku3A	83.33	58.06	39.34	83.33	67.74	45.9
1kw4A	92.86	71.43	51.43	92.86	71.43	52.86
1lm4A	89.47	67.37	48.15	78.95	66.32	51.85
1lo7A	100	97.14	84.29	100	100	92.14
1lpyA	75	49.38	35.19	75	54.32	35.8
1m4jA	48.15	52.24	47.37	48.15	56.72	50.38
1m8aA	66.67	38.71	26.23	66.67	41.94	34.43

1mk0A	100	83.67	59.79	100	75.51	52.58
1mugA	96.97	84.34	69.7	100	92.77	73.94
1nb9A	93.1	86.49	72.11	100	87.84	68.71
1ne2A	97.14	73.86	43.75	94.29	80.68	47.16
1npsA	100	95.45	75	100	90.91	70.45
1nrvA	85	54	35	85	54	41
1ny1A	95.74	91.53	77.45	97.87	89.83	81.7
1o1zA	100	95.58	77.88	100	93.81	80.53
1p90A	96	88.71	62.6	100	90.32	66.67
1pchA	100	100	87.5	100	100	86.36
1pkoA	100	96.77	80.65	100	95.16	82.26
1qf9A	100	93.81	75.77	100	97.94	76.29
1qjpA	100	88.41	61.31	100	95.65	72.99
1ql0A	87.5	61.16	45.23	95.83	68.6	50.62
1r26A	100	98.25	78.76	100	96.49	81.42
1roaA	100	89.29	77.48	100	92.86	76.58
1rw1A	100	73.68	57.89	100	91.23	70.18
1rw7A	97.87	88.14	74.47	100	97.46	85.11
1rybA	100	93.55	80.11	100	97.85	87.1
1smxA	88.24	84.09	63.22	100	88.64	63.22
1svyA	95	78.43	59.41	100	86.27	66.34
1t8kA	100	92.31	61.04	100	94.87	59.74
1tifA	93.33	73.68	55.26	93.33	73.68	48.68
1tqgA	100	83.02	59.05	100	92.45	63.81
1tqhA	100	94.21	76.03	95.83	90.08	79.75
1tzvA	100	87.32	63.12	100	91.55	63.12
1vfyA	76.92	61.76	40.3	92.31	61.76	40.3
1vhuA	100	93.75	83.85	100	94.79	84.9
1vjkA	100	97.73	87.5	100	90.91	77.27
1vmbA	100	100	88.79	100	100	90.65
1vp6A	100	98.51	84.96	100	100	89.47
1w0hA	92.5	83	68	90	85	70.5
1whiA	75	63.93	46.72	83.33	67.21	50
1wjxA	100	80.36	65.18	100	80.36	66.07
1wkcA	94.12	78.57	60.12	97.06	90.48	71.43
1xdzA	97.92	87.39	80.67	97.92	93.28	82.77
1xffA	100	86.55	78.15	100	99.16	88.24
1xkrA	95.12	83.5	62.93	95.12	88.35	68.29
2arcA	100	95.06	75.78	100	96.3	74.53
2cuaA	100	95.08	80.33	100	95.08	80.33
2hs1A	85	52	39.39	90	78	54.55
2mhrA	95.83	71.19	46.61	75	74.58	50.85
2phyA	40	25.4	20	40	30.16	24
2tpsA	95.56	88.5	77.43	100	94.69	78.76
2vxnA	88	76.8	67.87	88	84.8	73.9
3borA	97.44	91.75	73.71	100	91.75	76.8
3dqgA	93.33	83.78	63.51	96.67	87.84	67.57
5ptpA	100	86.49	72.52	100	97.3	85.59

Glossary of terms

OLS: The method of least squares is about estimating parameters by minimizing the squared discrepancies between observed data, on the one hand, and their expected values on the other.

L2 norm: L2-norm is also known as least squares. It is basically minimizing the sum of the square of the differences **(S)** between the target value ($\mathbf{Y_i}$) and the estimated values $\mathbf{f(x_i)}$.

L2 norm produces non-sparse coefficients. Sparsity refers to that only very few entries in a matrix is non-zero. In machine learning terms suppose the model has 100 coefficients but only 10 of them have non-zero coefficients, this is effectively saying that "the other 90 predictors are useless in predicting the target values".

So L2 norm is a soft constraint that minimizes the occurrence of the zero coefficients but just reduces them to weak predictors.

Non-invertible matrix: Any matrix with determinant zero is non-invertible. This is also called as singular matrix.

Determinant: In linear algebra, the determinant is a scalar value that can be computed from the elements of a square matrix and encodes certain properties of the linear transformation described by the matrix. The determinant of a matrix A is denoted det(A), det A, or |A|.

In the case of a 2 × 2 matrix the determinant may be defined as

$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

Maximum likelihood estimation: Since the data points $(X_1, y_1),.....,(X_N,y_N)$ are independently generated, the probability of getting all the y_n's in the data set from the corresponding X_n's would be the product

$\Pi\ P(yn|xn)$ for n =1 to N

log likelihood function: The log-likelihood is the natural logarithm of the likelihood.

Independent identical distribution: A collection of random variables is independent and identically distributed if each random variable has the same probability distribution as the others and all are mutually independent. This is abbreviated as i.i.d.

Convex function: A function $f : R^n \rightarrow R$ is convex if its domain is a convex set and for all x, y in its domain, and all $\lambda \in [0, 1]$, we have $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$.

Eigen vectors and Eigen Values: Eigen vector of a matrix A is a vector represented by a matrix X such that when X is multiplied with matrix A, then the direction of the resultant matrix remains same as vector X. Mathematically, above statement can be represented as:

$AX = \lambda X$, where A is any arbitrary matrix, λ are eigen values and X is an eigen vector corresponding to each eigen value.

Eigen decomposition: One of the most widely used kinds of matrix decomposition is called eigen decomposition, in which we decompose a matrix into a set of eigenvectors and eigenvalues