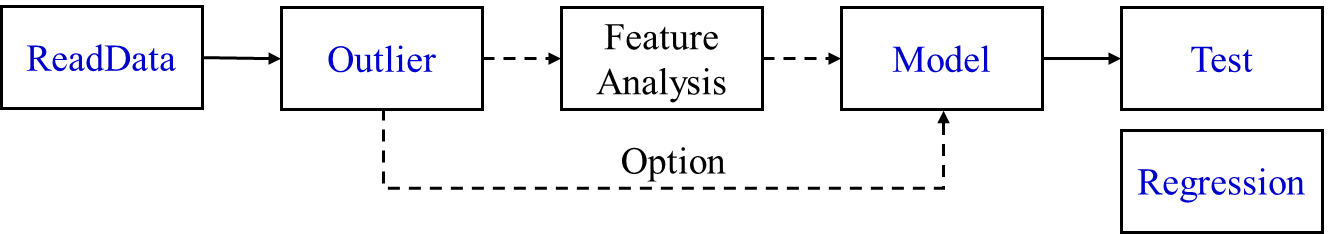
# 0. Introduction

The relationships of five page are as follows:



ReadData will get data and visualize it.

Outlier will select outliers in data, delete them and visualize them.

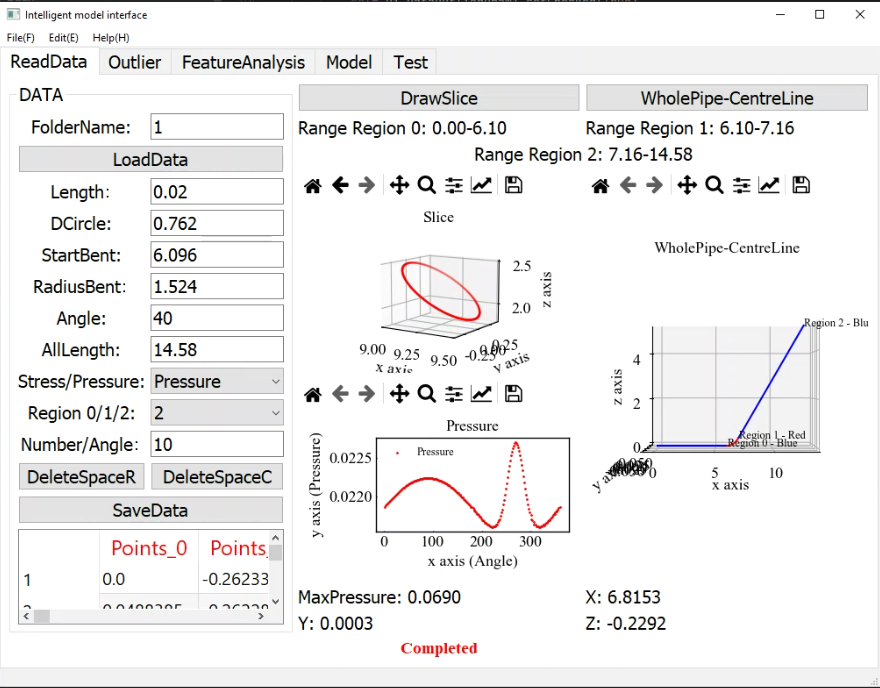
FeatureAnalysis will compute the relation of features and visualize it. Besides, we can normalize the data in this page.

Model will build intelligence model, then train and verify it.

Test will test the training model and visualize the results.

Regression will perform a formula regression of the maximum value of the variable using the radius and Angle of the bend.

## 0.1 ReadData



There are 6 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

*LoadData*: Back-end function, "DateInputBtn", It will open the file selection screen, read the selected file and display it.

*Stress/Pressure:* Input data type

*Region:* Select visualization area for plotting

*DeleteSpaceR:* Back-end function, " btnDelRows", It will delete empty row in data.

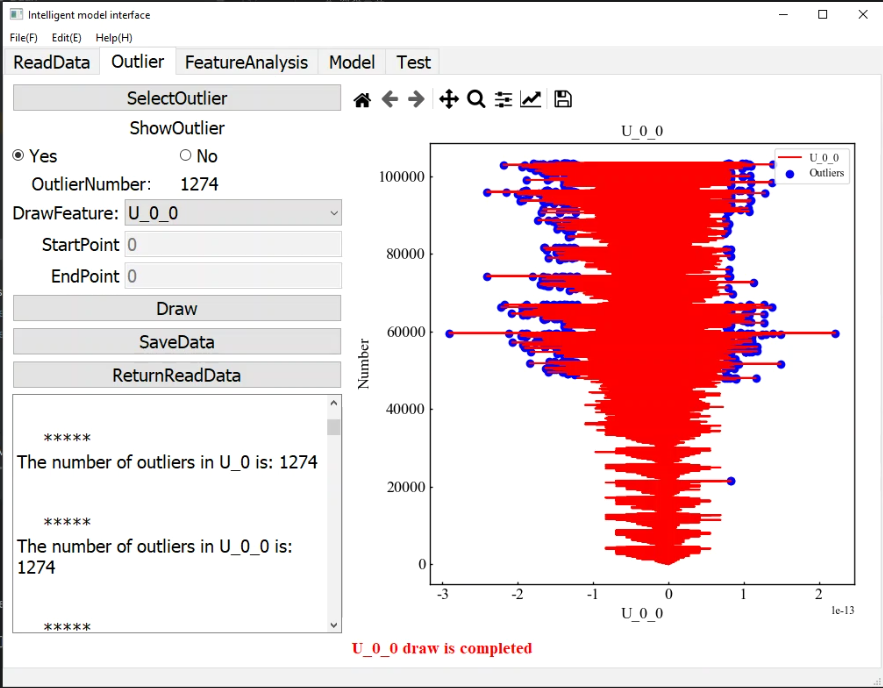
*DeleteSpaceC:* Back-end function, " btnDelCols", It will delete empty column in data.

*SaveData*: Back-end function, " btnSaveData", The corrected data presentation will be modified

*WholePipe-CentreLine*: Back-end function, "DrawFig", Draws a slice of the selected location (as determined by the input page data).

*DarwSlice*: Back-end function, " DrawWholePipe ", Draw the center curve of the pipe.

## 0.2 Outlier



There are 3 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

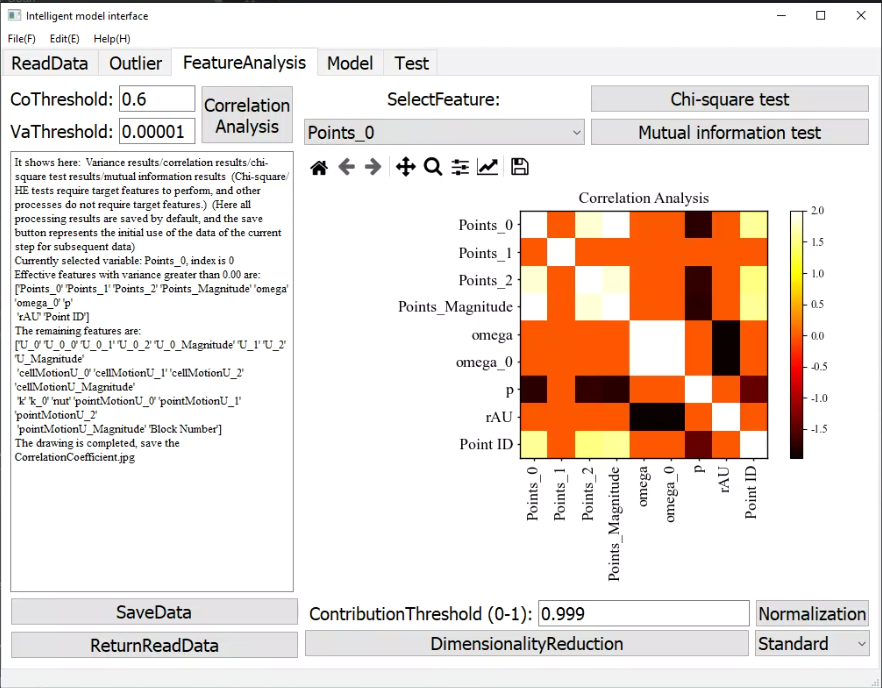
*SelectOutlier*: Back-end function, " Outlier", The modified 3 sigma is used to determine outliers for each feature in the data.

*Draw*: Back-end function, "OutlierDraw", Plots values and outliers for selected features.

*SaveData*: Back-end function, " SaveDataBtn", The corrected data presentation will be modified

*ReturnReadData*: Return to ReadData page.

## 0.3 FeatureAnalysis



There are 6 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

*CorrelationAnalysis*: Back-end function, " CorrDraw", Pearson correlation coefficient and Spearman correlation coefficient are used to calculate the correlation coefficient between each feature.

*Chi-square test*: Back-end function, " chi2Btn", Chi-square test is used to analyze the relationship between features.

*Mutual information test*: Back-end function, " mutualInforBtn", The relationship between features is analyzed by mutual information method.

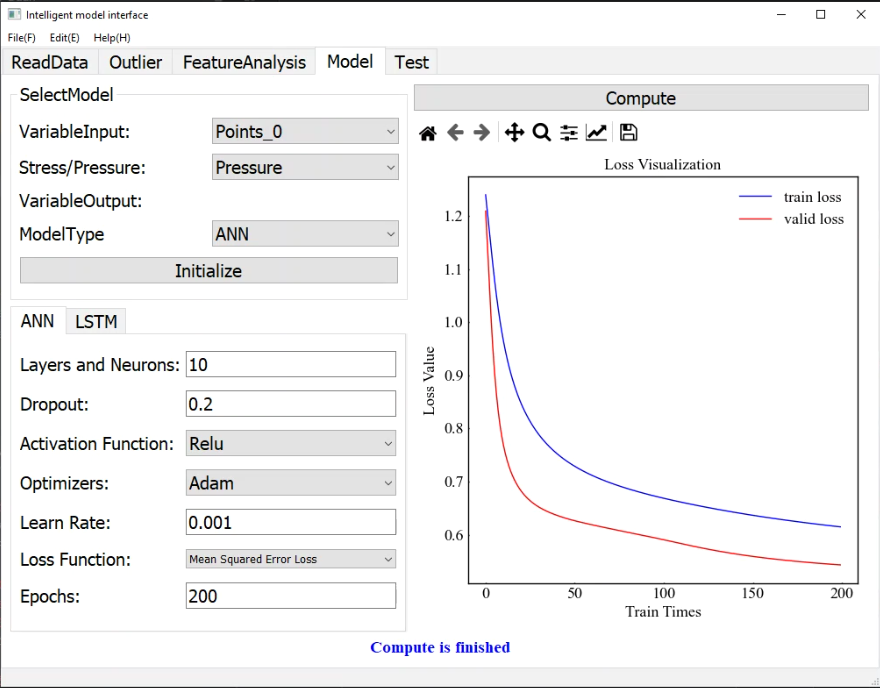
*Normalization*: Back-end function, " NondimenBtn", The data is processed without dimension. It can be divided into standardization method and maximum and minimum method.

*DimensionalityReduction*: Back-end function, " DimenComBtn ", The principal component analysis is used to process the data and reduce the data dimension. (Processed data will lose its physical meaning).

*SaveData*: Back-end function, " SaveData", The corrected data presentation will be modified

*ReturnReadData*: Return to ReadData page.

## 0.4 Model



There are 2 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

*ANN/LSTM*: Select type of model for training.

*Initialize*: Back-end function, " Initialize", Since there are many parameters of the intelligent model, the parameters of the model can be quickly initialized through this function (the initial parameters are only for reference and are not necessarily applicable to the current data).

-------------------------------------------------------

*Layers and Neurons:* The number of network layers and the number of neurons, for example, 10-3 indicates that in addition to the input and output layers, there are two hidden layers, and the number of neurons is 10 and 3 respectively

*Dropout:* Prevent overfitting, range 0-1

Activation Function: Activation function

*Optimizer:* Neuron optimization method

*Learn Rate:* Optimizer learning speed

*Loss Function:* Training loss value calculation method

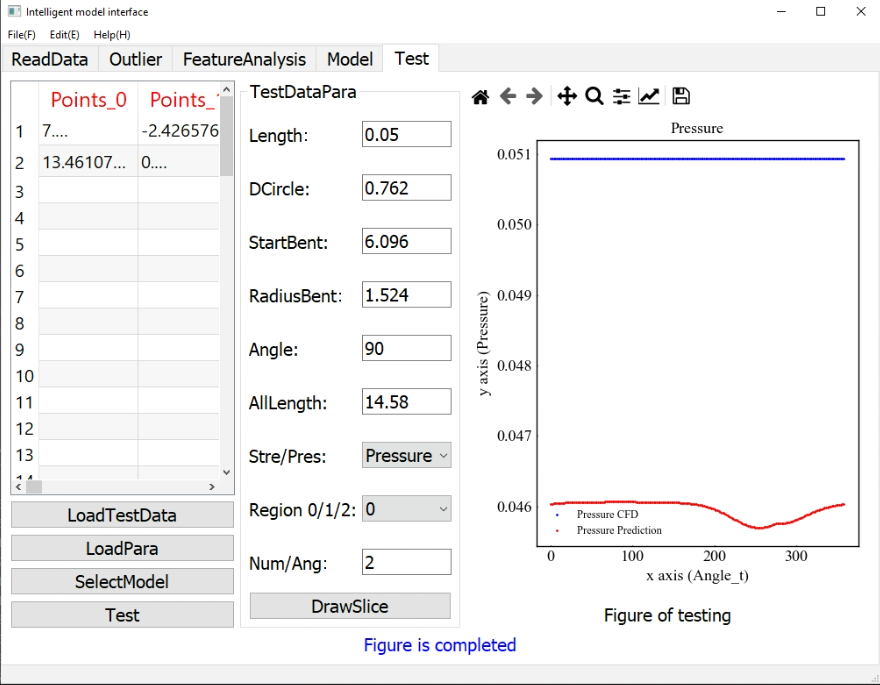
*Epochs:* Number of training

(For details about functions in each part, please refer to 3.Model building)

------------------------------------------------------------

*Compute*: Back-end function, " Compute", The data is divided into validation set and training set by 7:3. The model is trained and verified according to the set parameters.

## 0.5 Test



There are 2 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

*LoadTestData*: Back-end function, " LoadTestData", Select the file window. Select the data to test the model against.

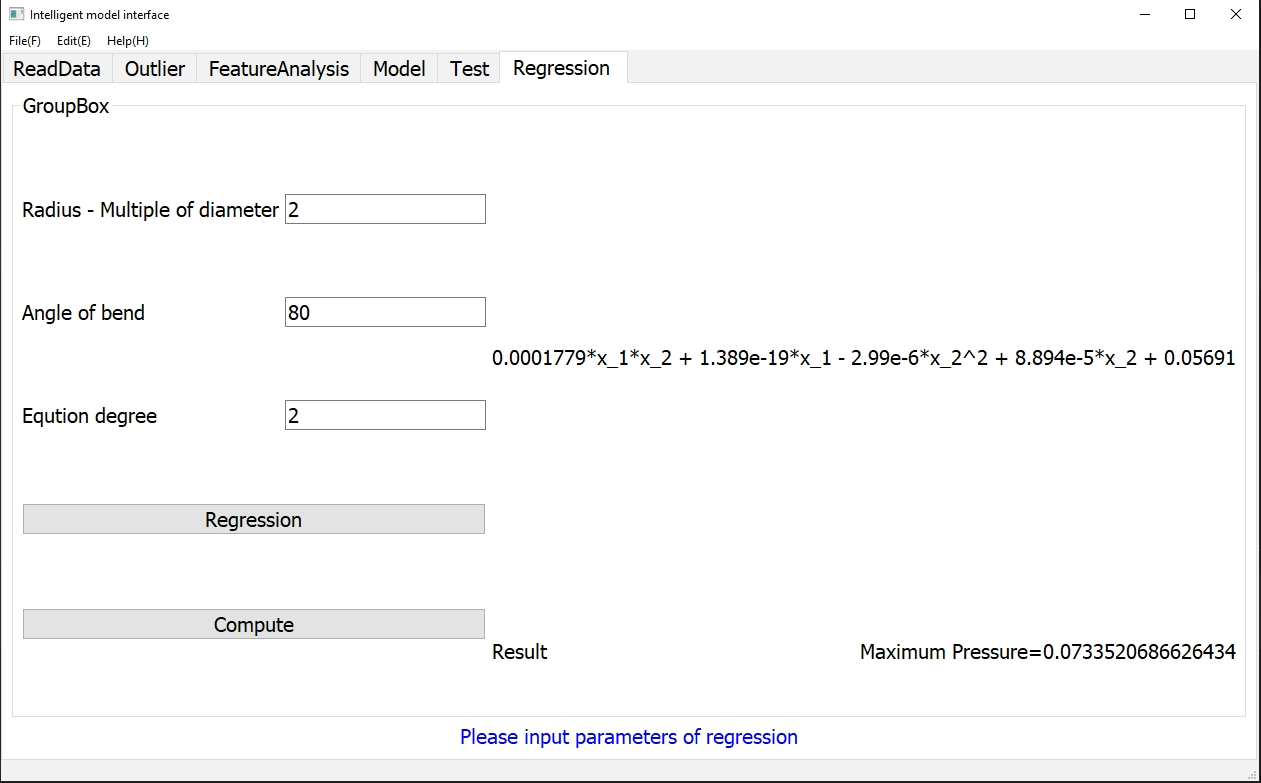
*LoadPara*: Back-end function, " LoadParaBtn", Select the file window. Select model normalization parameters. See TestDataPara framework for slice location selection.

*SelectModel*: Select the file window. Select the model to be tested.

*Test*: Back-end function, " Test", Test button. When the test data is imported and the normalized parameters and corresponding model are selected, the second button can be used to test the model.

*DrawSliceT*: Back-end function, " DrawSliceT", Draw the simulation results of slices and the prediction results of intelligent models. See TestDataPara framework for slice location selection.

## 0.6 Regression



There are 2 functions. The relationship between buttons and functions is as follows (*Italics* represent buttons):

*Regression*: Back-end function, "Regression", Select the folder and perform polynomial regression on the data in the folder. The folder name must be Variable\_Radius\_Degree. The Degree of the regression polynomial is determined by the interface degree.

*Compute*: Back-end function, "ComputeR", Calculate the maximum value of the variable under the new radius and Angle by the regression polynomial.

-------------------------------------------------------

*Raduis – Multiple of diameter:* The radius of the bend is expressed as a multiple of the diameter of the pipe

*Angle of bend:* The maximum Angle of the bend

*Eqution degree:* The highest degree of polynomial regression. (*Regression*)

------------------------------------------------------------

# 1. Outlier

This method is based on the properties of the normal distribution. The area within the horizontal axis interval (μ − 3 σ , μ + 3 σ ) of a normal distribution is 99.730020%. If it does not obey the normal distribution, it can be described using the principal n times the standard deviation. The specific n is determined by the application scenario.







It has also been corrected. When continuous outliers occur, outlier calibration is cancelled.

# 2. Feature Analysis

## 2.1. Pearson correlation coefficient

Pearson Correlation is a way to measure the similarity of vectors. The output range is -1 to +1, where 0 represents no correlation, negative values represent negative correlation, and positive values represent positive correlation. The Pearson correlation coefficient is optimized on the Euclidean distance and the vector values are centered. That is, the average value of the elements is subtracted from all dimensions in the two vectors. After centering, the average value of all dimensions is basically 0. Then calculate the cosine distance from the centering result, but the calculation of the cosine distance requires that all values in each vector must be non-empty. If two vectors v1=(3,2,4), v2=(-1,2,null ), the cosine distance calculation cannot be performed. The Pearson correlation coefficient assigns all null dimensions in the vector to 0, and then performs cosine calculation on the result. The calculation formula of Pearson correlation coefficient is as follows:



Assuming there are two variables X and Y, then the application of Pearson correlation coefficient is understood as:

(1) When the correlation coefficient is 0, the X variable and the Y variable are not related;

(2) When the value of X and the value of Y increase or decrease at the same time, the two variables are positively correlated, and the correlation coefficient is between 0 and 1;

(3) When the value of X increases and the value of Y decreases, or when the value of X decreases and the value of Y increases, the two variables are negatively correlated, and the correlation coefficient is between -1 and 0.

The larger the absolute value of the correlation coefficient, the stronger the correlation. The closer the correlation coefficient is to 1 or -1, the stronger the correlation. The closer the correlation coefficient is to 0, the weaker the correlation. Usually, the correlation strength of variables is judged through the following value range:

1. 0.8-1.0 Very strong correlation
2. 0.6-0.8 strong correlation
3. 0.4-0.6 Moderate correlation
4. 0.2-0.4 weak correlation
5. 0.0-0.2 Very weak phase or no correlation

## 2.2. Spearman correlation coefficient

Spearman rank correlation is mainly used to solve problems related to name data and ordinal data. It is suitable for data with two columns of variables and linear relationship with the nature of hierarchical variables. Derived by the British psychologist and statistician Spearman based on the concept of product-difference correlation, some people regard Spearman rank correlation as a special form of product-difference correlation.

Spearman's correlation coefficient is also commonly called Spearman's rank correlation coefficient. "Rank" can be understood as a kind of order or sorting, then it is solved based on the sorting position of the original data. If there are two sets of data, X and Y, the Spearman (rank) correlation coefficient calculation formula is:



A Spearman correlation coefficient >0 indicates a positive correlation; a Spearman correlation coefficient <0 indicates a negative correlation. The closer to 1 and -1 the stronger the correlation. A Spearman correlation coefficient of zero indicates that there is no trend in Y as X increases.

## 2.3. Chi-square Test

The chi-square test is a commonly used hypothesis testing method. Its null hypothesis H0 is: there is no difference between the observed frequency and the expected frequency. The so-called null hypothesis, also known as the null hypothesis, is generally a hypothesis that is intentionally overturned in the hope of proving that it is wrong. For example, if we want to prove that two obviously related variables are related, we can make a null hypothesis H0: assuming that the two are unrelated. Usually, the chi-square test is a test of independence of discrete variables. The null hypothesis of the chi-square test is that two discrete variables are independent of each other. In feature selection, it is used to distinguish labels and features. It can be used to determine whether a feature and a label are independent. If so, it means that the feature is not helpful in predicting the label. Therefore, in many cases, the chi-square test is a very important method to eliminate irrelevant features.

The basic idea of the chi-square test:

First, assume that H0 is established. Based on this premise, the chi-square value is calculated, which represents the degree of deviation between the observed value and the theoretical value. Then based on the chi-square distribution and degrees of freedom, the probability P of obtaining the current statistics and more extreme situations can be determined when the H0 hypothesis is established. If the P value is very small, it means that the observed value deviates too much from the theoretical value, and the null hypothesis should be rejected, indicating that there is a significant difference between the variables; otherwise, if the null hypothesis is accepted, it cannot be considered that the actual situation represented by the sample is different from the theoretical hypothesis. As for how large the chi-square must be in each specific study to reject H0, it must be determined by using the chi-square distribution to find the corresponding P value.

Calculation and significance of the chi-square value: The chi-square value represents the degree of deviation between the observed value and the theoretical value. The basic idea of calculating this degree of deviation is as follows. Let A represent the observed frequency of a certain category, E represent the expected frequency calculated based on H0, and the difference between A and E is called the residual. Obviously, the residual can represent the degree of deviation between the observed value of a certain category and the theoretical value. However, if the residuals are simply added up to express the difference between the observed frequency and the expected frequency of each category, there are certain shortcomings. Because there are positive and negative residuals, they will cancel each other out after being added, and the sum will still be 0. On the other hand, residual size is a relative concept. Relative to the expected frequency of 10, the residual of the expected frequency of 20 is very large, but relative to the expected frequency of 1,000, the residual of 20 is very small. With this in mind, one divides the squared residuals by the expected frequency and sums them to estimate the difference between the observed frequency and the expected frequency.



## 2.4. Mutual information method

Mutual information Mutual information of two random variables is a measure of the interdependence between the variables. Mutual information measures the information shared by two random variables - knowing the random variable X, the degree to which the uncertainty about the random variable Y is reduced (or knowing the random variable Y, the degree to which the uncertainty about the random variable (X;Y) represents.

The random variable X represents the number of points thrown by a balanced six-sided die, and Y represents the parity of X. Here we assume that when X is an even number, Y=0; when X is an odd number, Y=1. If we know X, such as X=1, we can determine Y=1. (The possibility of losing the information that Y=0, the uncertainty information of Y is reduced). Similarly, if we know Y=0, we can determine X=2 or 4 or 6. (The possibility of losing the information that X=1 or 3 or 5, the uncertainty information of X is reduced). Therefore, we say that there is mutual information between random variables X and Y.

The definition of mutual information:



## 2.5 Normalization

Dimensionless transformation converts data of different specifications into the same specification. Common dimensionless methods include standardization and interval scaling. The premise of standardization is that the feature values obey the normal distribution. After standardization, they are converted into standard normal distribution. The interval scaling method uses boundary value information to scale the feature value interval to the range of a certain feature, such as [0, 1], etc.

Standardization processes data according to the columns of the feature matrix. It converts the feature values of the sample into the same dimension by finding the z-score method. Normalization is to process data according to the rows of the feature matrix. Its purpose is to have a unified standard for sample vectors when calculating similarity in dot multiplication operations or other kernel functions, that is to say, they are all converted into "unit vectors".

(1) Standardization

Standardization requires calculating the mean and standard deviation of the features, and the formula is expressed as:

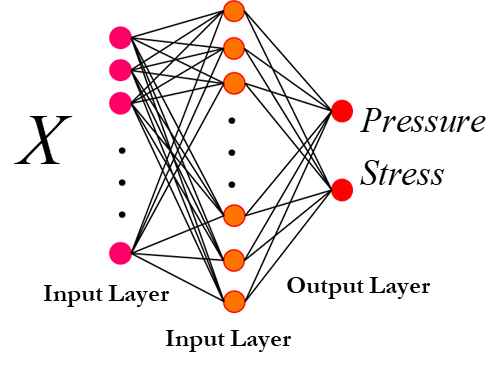


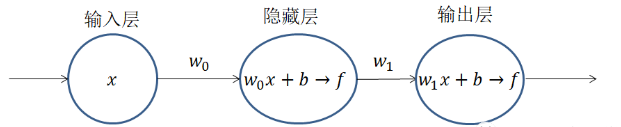
(2) Normalization

There are many ideas for the interval scaling method. A common one is to use the two maximum values for scaling, that is, the max-min method. The formula is expressed as:



# 3. Model building





Schematic diagram of fully connected neural network

Not counting the input layer, the above network structure has a total of two layers, the hidden layer and the output layer. The calculations in their "circles" are the calculation combinations in the figure below (the activation function is not unique, this example is  the sigmoid activation function, the following example The representative symbol when it is a sample)



Each level uses the output of the previous level as input. After  calculation and  operation, the purpose of  the operation is to compress the output value range to (0, 1), which is the so-called normalization, because each level The output values will be used as the input of the next level. Only by normalizing the input can we avoid an input being infinite, causing other inputs to be invalid and output to the next level. For the sake of convenience in this article, when passing the activation function  as input to the next neural layer below, it is still written as .

The ANN algorithm is a kind of supervised learning, that is, the input X has a corresponding real value Y, and the loss between the output Y of the neural network and the real value Y is the network back propagation. The basic idea of backpropagation is to adjust the network parameters by calculating the error between the output layer and the expected value, thereby making the error smaller. The training process of the entire network is the process of continuously reducing the loss. The expression is:



After simplification of the above formula (this example uses the least squares method, there are also many other algorithms, such as maximum likelihood estimation, etc.), we can see  that they are all constant coefficients,  and  the unknown is the sum. That is, in order to minimize , we require Solve for the best  and .

The ANN algorithm uses the gradient descent method to solve  and , which requires extreme values. First, find the derivatives, and then find the partial derivatives of  and  respectively to determine the direction of reduction, that is, the gradient:



In order to find the optimal solution, the weights and biases should be reduced along the negative gradient direction, that is:



The same is true for multiple samples, but when multiple samples are input at the same time, special attention needs to be paid to dimension matching. At the same time, for the loss function.

It can be found that some parameters of the ANN neural network model need to be given by the designer, and some parameters must be solved by the model itself. Learning rate, number of hidden layers, number of neurons in each hidden layer, selection of activation function, selection of loss function (cost function), etc. These parameters need to be set manually and are called hyperparameters. Other parameters, such as weight matrices and bias coefficients, can be obtained through model calculations after determining the hyperparameters. These parameters are called ordinary parameters, or parameters for short.

In fact, it is very difficult to determine the hyperparameters because it is difficult to know what hyperparameters will make the model perform better. For example, a learning rate that is too small may cause the model to converge too slowly, and a learning rate that is too large may cause the model to not converge; another example is the design of the loss function. If the loss function is not designed well, the model may not converge; another example is , when there are too many layers, how to design the network structure to avoid gradient disappearance and gradient explosion, etc.

## 3.1. Introduction to activation function

Here we introduce several common activation functions and loss functions to provide a basis for algorithm verification.

The main role of the activation function is to provide the nonlinear modeling capability of the network. It has the following characteristics: Differentiability: When the optimization method is based on gradients, this property is necessary; Monotonicity: When the activation function is monotonic, the single-layer network can guarantee a convex function; The range of the output value: When the activation function When the function output value is limited, the gradient-based optimization method will be more stable, because the representation of features is more significantly affected by the limited weight value; when the output of the activation function is infinite, the model training will be more efficient, but in This situation is small and generally requires a smaller learning rate.

1) Sigmoid function



The sigmoid function used to be used a lot, but in recent years, fewer and fewer people use it. Mainly because of its inherent shortcomings. In deep neural networks, gradient backpropagation leads to gradient explosion and gradient disappearance. The probability of gradient explosion is very small, while the probability of gradient disappearance is relatively high. The output of Sigmoid is not zero mean. Its analytical formula contains power operations, which is relatively time-consuming to solve by computer.

2) tanh function



It solves the problem of non-zero-centered output of the Sigmoid function. However, the problem of gradient vanishing and the problem of power operation still exist.

3）ReLU, Leaky-ReLU function





The full name of ReLU is Rectified Linear Units, which is an activation function that appeared later. It can be seen that when x<0, ReLU is hard saturated, and when x>0, there is no saturation problem. Therefore, ReLU can keep the gradient from decaying when x>0, thereby alleviating the vanishing gradient problem. This allows us to directly train deep neural networks in a supervised manner without relying on unsupervised layer-by-layer pre-training.

However, as training progresses, some inputs will fall into the hard saturation zone, causing the corresponding weights to fail to be updated. This phenomenon is called "neuronal death." Similar to sigmoid, the output mean of ReLU is also greater than 0. The offset phenomenon and neuron death will jointly affect the convergence of the network.

For the hard saturation problem at x<0, we respond to ReLU

4) ELU function



Fusion of sigmoid and ReLU, with soft saturation on the left and no saturation on the right. The linear part on the right side allows the ELU to alleviate gradient disappearance, while the soft saturation on the left side allows the ELU to be more robust to input changes or noise. A small problem with it is that it is slightly more computationally intensive. Similar to Leaky ReLU, although it is better than ReLU in theory, there is currently no good evidence in actual use that ELU is always better than ReLU.

## 3.2 Introduction to loss function

The loss function, also called the cost function, is used to evaluate the degree of inconsistency between the predicted value of the model and the true value. It is also the objective function of optimization in the neural network. The process of neural network training or optimization is to minimize the loss. In the process of function, the smaller the loss function, the closer the predicted value of the model is to the true value, and the better the robustness of the model.

1) Square loss function



The squared loss function refers to the square of the difference between the predicted value and the true value. The greater the loss, the greater the difference between the predicted value and the true value. The square loss function is mostly used in linear regression tasks.

2) Hinge loss function



The Hinge loss function is usually suitable for binary classification scenarios and can be used to solve the problem of interval maximization. It is often used in the famous SVM algorithm.

3) Logarithmic loss function



 is the category of the known classification,  is the sample value. We need to make the probability reach the maximum value. That is to say, we require a parameter value that maximizes the probability value of the current set of data output.

The logarithmic loss function is also a common loss function and is often used in logistic regression problems.

4) Cross entropy loss function



The cross-entropy loss function is essentially a logarithmic loss function and is often used in multi-classification problems. The cross-entropy loss function is often used when the sigmoid function is used as the activation function, because it can perfectly solve the problem of slow update of the weight of the square loss function.

5) Log-likelihood loss function



In deep learning, the log-likelihood function is often used with the softmax activation function.

## 3.3 Introduction to optimizer

In machine learning, there are many optimization methods to try to find the optimal solution for the model. For example, the most basic gradient descent method can be used in neural networks.

The gradient descent method is the most basic type of optimizer. Currently, it is mainly divided into three gradient descent methods: standard gradient descent method (GD), stochastic gradient descent method (SGD) and batch gradient descent method (BGD).

1) Standard gradient descent method (GD)

Assume that the model parameters to be learned and trained are , and the loss function is , then the partial derivative of the cost function with respect to the model parameters, that is, the correlation gradient is , and the learning rate is , then the gradient descent method is used to update the parameters as:



The update and adjustment of model parameters is related to the gradient of the cost function with respect to the model parameters, that is, the model parameters are continuously reduced along the direction of the gradient, thereby minimizing the cost function. The basic strategy can be understood as "finding the fastest path down the mountain within a limited sight distance".

The standard gradient descent method has two main disadvantages: 1. Slow training speed 2. Easy to fall into the local optimal solution

2) Batch Gradient Descent (BGD)

Assume that the total number of batch training samples is , the input and output samples of each time are, the model parameters are, the cost function is, the gradient of the cost function for each input sample is , the learning rate is , then use the batch gradient descent method to update the parameter expression for:



The adjustment and update of model parameters is related to the sum of the cost functions of all input samples (i.e., batch/global error). That is, each weight adjustment occurs after batch samples are input, instead of updating the model parameters every time a sample is input. This will greatly speed up training.

The batch gradient descent method takes less time to train than the standard gradient descent method, and the direction of each descent is correct.

3) Stochastic Gradient Descent (SGD)

Compare the batch gradient descent method, assuming that a sample is is randomly selected from a batch of training samples n. The model parameter is W, the cost function is J(W), the gradient is ΔJ(W), and the learning rate is . Then the parameter expression updated using the stochastic gradient descent method is:



Although randomness and noise are introduced here, the expectation is still equal to correct gradient descent. Although SGD requires many steps, it has very low requirements on gradients (calculating gradients is fast). Training is fast when applied to large data sets.

In addition, there are two methods of momentum optimization for SGD: Momentum and NAG

The main idea of using the stochastic gradient descent method (SGD) of momentum is to introduce a momentum of accumulated historical gradient information to accelerate SGD.

Take a small batch sample  of size from the training set, and the corresponding real values are , then the Momentum optimization expression is:



In order to mainly solve two problems of SGD: one is the stochastic gradient method (introduced noise); the other is the ill-posed problem of Hessian matrix

The Nesterov accelerated gradient (NAG) algorithm is a variant of the Momentum momentum algorithm. The updated model parameter expression is as follows:



The Nesterov momentum gradient is calculated after the current velocity is applied to the model parameters, so it can be understood as adding a correction factor to the standard momentum.

4) Adaptive learning rate optimization algorithm

The adaptive learning rate optimization algorithm is aimed at the learning rate of the machine learning model. The traditional optimization algorithm either sets the learning rate to a constant or adjusts the learning rate according to the number of training times. The possibility of other changes in the learning rate is greatly ignored. However, the learning rate has a significant impact on the performance of the model, so some strategies are needed to find ways to update the learning rate to increase the training speed.

The current adaptive learning rate optimization algorithms mainly include: AdaGrad algorithm, RMSProp algorithm, Adam algorithm and AdaDelta algorithm.

The AdaGrad algorithm adapts the learning rate of all model parameters independently, scaling each parameter inversely as the square root of the sum of its historical averages of all gradients. Parameters with the largest gradients of the cost function will correspondingly have a rapidly decreasing learning rate, while parameters with small gradients will have a relatively small decrease in the learning rate.

The AdaGrad algorithm optimization strategy can generally be expressed as:



The main advantage of Adagrad is that it does not require manual adjustment of the learning rate, it can be adjusted automatically; the disadvantage is that as the number of iterations increases, the learning rate will become smaller and smaller, and will eventually approach 0.

The general strategy of RMSProp algorithm can be expressed as:



The RMSProp algorithm has been empirically proven to be an effective and practical deep neural network optimization algorithm. It is currently one of the optimization methods often adopted by deep learning practitioners.

Both the AdaGrad algorithm and the RMSProp algorithm need to specify the global learning rate, and the AdaDelta algorithm combines the update step size of each parameter of the two algorithms.

In the early and middle stages of model training, AdaDelta performs very well, with good acceleration and fast training speed. In the later stages of model training, the model will repeatedly jitter around local minima.

Momentum in Adam is directly incorporated into the estimation of the first moment of the gradient (exponentially weighted). Secondly, compared to RMSProp, which lacks a correction factor and causes the second-order moment estimate to be highly biased in the early stages of training, Adam includes a bias correction that corrects the first-order moment (momentum term) initialized from the origin and the (non-central) second-order moment. Order moment estimation.

Currently, the most popular and highly used optimizers (algorithms) include SGD, SGD with momentum, RMSprop, RMSProp with momentum, AdaDelta, and Adam.