ECE 449 Machine Learning (21Fa): Assignment 1

Xu Ke 3190110360

Sep. 26, 2021

1 Problem 1: Basic definitions

1.1 the difference between regression and classification

The main difference between *Regression* and $Classification$ is the type of output variable: the quantitative output is called *Regression*, which is continuous variable prediction; while the qualitative output is called $Classification$, which is discrete variable prediction. More specifically, *Regression* problems are usually used to predict a value, such as housing price, future weather conditions, etc. For example, the actual price of a product is 408 yuan, and the predicted value is 449 yuan through regression analysis. A more common regression algorithm is the linear regression algorithm (LR). In addition, when regression analysis is used in neural network, $Softmax$ function does not need to be added to the top layer, but can be directly added to the previous layer. Regression is an approximate prediction of the true value. However, $Classification$ problems are used to label things, usually resulting in discrete values. For example, to determine whether an animal on a picture is a cat or a dog, classification is usually based on regression, and the last layer of classification is usually determined by the $Softmax$ function. Classification does not have the concept of approximation, the final correct result is only one. The most common classification method is logistic regression, or logical classification.

1.2 the difference between classification and clustering

Clustering is an unsupervised learning technique used to group similar instances on the basis of features while $Classification$ is a supervised learning technique used to assign predefined tags to instances on the basis of features. More specifically, *Clustering* will not use training sets, but use statistical concepts, and datasets without labels will be split into subsets with similar features, in order to find whether there is any relationship between them. However, $Classification$ will use training sets to find similarities, and will use algorithms to categorize the new data according to the observations of the training set with labels for some points, in order to find which class a new object belongs to from the set of predefined classes. In my opinion, *Unsupervised Learning* is more suitable for *Clustering*.

2 Problem 2: Basic definitions

2.1 active learning

Active learning is a method of machine learning (or more specifically, iterative supervised learning) in which a learning algorithm can interactively query a user to label new data points with the desired outputs. Considering such situations where unlabeled data is abundant but manual labeling is expensive, learning algorithms can actively query the user for labels.

2.2 incremental learning

Incremental Learning is a method of machine learning in which data is continuously used to extend the existing model's knowledge (i.e. to further train the model) and it represents a dynamic technique of supervised learning and unsupervised learning, in order to adapt to new data without forgetting its existing knowledge (i.e. faster classification or forecasting times).

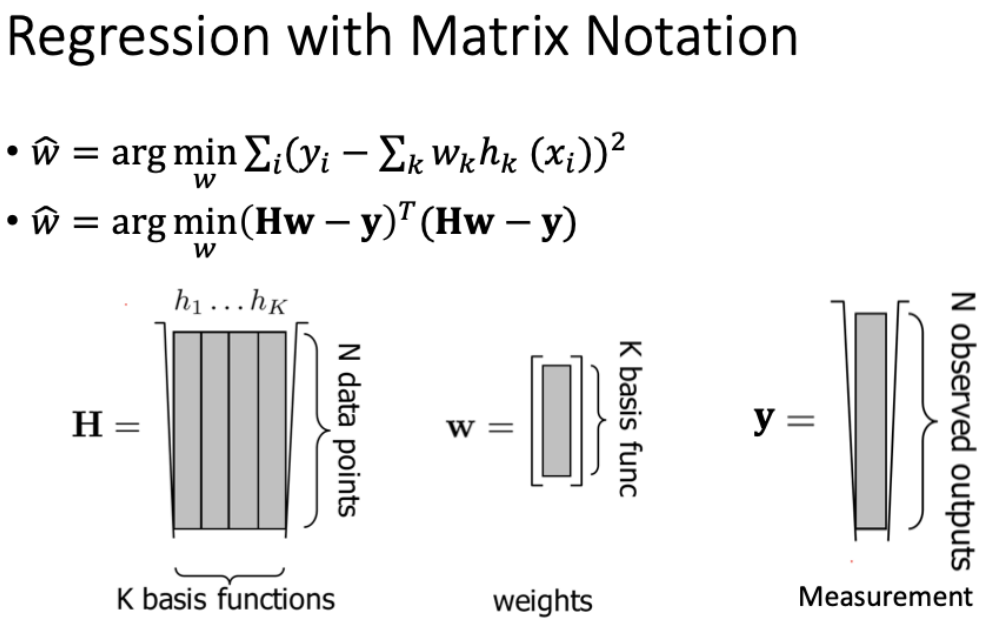
2.3 curriculum learning

Curriculum Learning is a method of machine learning in which researchers formalize the training strategy that Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. And it can be seen as a particular form of continuation method (a general strategy for global optimization of non-convex functions), according to Prof. Yoshua Bengio.

2.4 federated learning

Federated Learning is a method of machine learning which trains an algorithm across multiple decentralized edge devices or servers holding local data samples without exchanging them and it enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data.

3 Problem 3: Matrix Notation



According to Lecture 1 Slides 10 shown as the Figure above, in this case we have two dimensional patterns (2,1),(3,5),(4,3),(5,6) and a function y = $w_1 + w_2 x + w_3 x^2$, here $h_0(x) = 1$, $h_1(x) = x_1$, $h_2(x) = x_2^2$. Thus, we will have:

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} 1 \\ 5 \\ 3 \\ 6 \end{pmatrix}$$

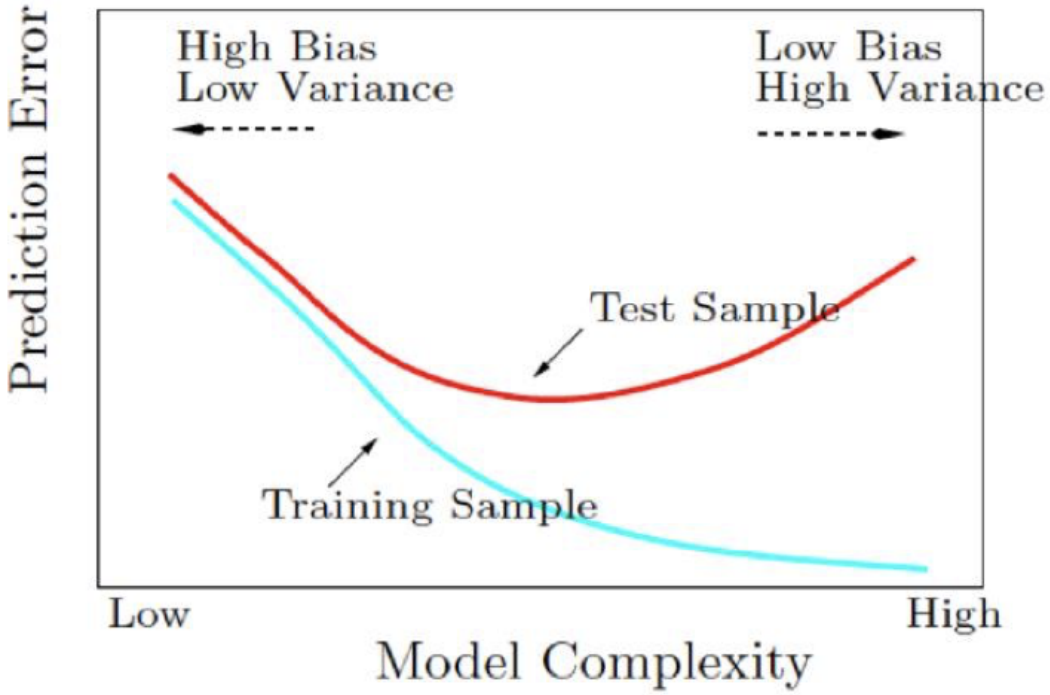
$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

4 Problem 4: Bias via Variance

4.1 the difference between bias and variance

Bias measures the degree to which the average estimation results of a learning algorithm can approximate the learning target and a high bias means a bad match. While *Variance* measures the degree of dispersion in the face of different training sets of the same size and a high variance means a weak match, and the data is scattered. To be more intuitive, given a target, the lower the bias is, the more sample will be distributed near the bull 's-eye, and the lower the variance is, the more sample will be concentrated somewhere.

4.2 explanation about why bias and variance change as model complexity changes based on the figure below



From the Figure 1 above we can easily know that for *Test Sample* if the model complexity is low, the prediction error is high consisting of high *Bias* and low *Variance*, while if the model complexity is high, the prediction error is high consisting of low *Bias* and high *Variance*. And for *Training Sample*, it is clear that the prediction error decreases as model complexity increases. The reason for this phenomenon is shown as following separately. When we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data, *Underfitting* will happen since models will be unable to capture the underlying pattern of the data. These models usually have high bias and low variance and are very simple to capture the complex patterns in data like Linear and logistic regression. In addition, *Overfitting* will happen when we train our model a lot over noisy dataset because our model will capture the noise along with the underlying pattern in data. These models have low bias and high variance and are very complex like Decision trees which are prone to overfitting.

5 Problem 5: Training data, Validation data and Test data

5.1 the difference between training data and test data

Training Data is the data used for learning, which is to fit the parameters of the classifier (i.e. weights). While *Test Data* is the data used only to assess the performance or generalization of a fully specified classifier.

5.2 explanation about why we need validation set

Most machine learning algorithms have hyperparameters that can be set to control the behavior of the algorithm. The values of hyperparameters are usually not learned by the learning algorithm itself. Sometimes an option is set to the hyperparameter of the learning algorithm because it is too difficult to optimize. More commonly, this option must be hyperparameters because it is not suitable for learning on a training set. This applies to all hyperparameters that control model capacity. If hyperparameters are learned on a training set, these hyperparameters always tend to the maximum possible model capacity, leading to overfitting. For example, higher degree polynomials and weight attenuation parameter settings always fit better on the training set than lower degree polynomials and positive weight attenuation settings. To solve this problem, we need a sample of the $Validation\ Set$ that the training algorithm cannot observe. A test set consisting of samples with the same distribution as the training data can be used to estimate the generalization error of the learner after the learning process is complete. The point is that test samples should not participate in any way in model selection, including the setting of hyperparameters. For this reason, samples from test sets cannot be used in $Validation\ sets$. Therefore, we always build $Validation\ Sets$ from training data. In particular, we divide the training data into two disjoint subsets and one of them is for learning parameters while the other is used as a $Validation\ Set$ to estimate generalization errors during or after training and update hyperparameters.

6 Problem 6: Ridge Regression

$$\hat{W}_{ridge} = arg\min_{w}(\mathbf{Hw}-\mathbf{y})^T(\mathbf{Hw}-\mathbf{y}) + \lambda \mathbf{w}^T I_{0+K}\mathbf{w} \tag{1}$$

From the equation mentioned above, it is clear that we can get the cost function for $RidgeRegression$, which is $(\mathbf{Hw}-\mathbf{y})^T(\mathbf{Hw}-\mathbf{y})+\lambda \mathbf{w}^T I_{0+K}\mathbf{w}$. And then since we want to minimize the cost function, with previous knowledge from Calculus class, we can easily take the derivative of this cost function and let it equals to 0 to reach critical point. Thus we can get:

$$\nabla cost(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{Hw}) + 2\lambda I\mathbf{w} = 0 \tag{2}$$

By taking division by 2 on the both sides of the equation and separate each term, we can get:

$$-\mathbf{H}^T\mathbf{y} + \mathbf{H}^T\mathbf{Hw}) + \lambda I\mathbf{w} = 0 \tag{3}$$

By extracting common factor of the second and third term, we can get:

$$(\mathbf{H}^T\mathbf{H}) + \lambda I)\mathbf{w} = \mathbf{H}^T\mathbf{y} \tag{4}$$

By multiplying the inverse matrix of $\mathbf{H}^T\mathbf{H} + \lambda I$, we can solve for $\hat{W}_{ridge}$:

$$\hat{W}_{ridge} = (\mathbf{H}^T\mathbf{H} + \lambda I_{0+K})^{-1}\mathbf{H}^T\mathbf{y} \tag{5}$$

Hence, the closed form solution proved.

7 Problem 7: LASSO and ridge regression

7.1 explanation about why we employ LASSO regression and ridge regression on the basis of original linear regression

The original *Linear Regression* is only applicable to low dimensions and variables cannot have multicolinearity. So *Ridge Regression* is designed to help solve multicolinearity. L2 penalty term is also added to make calculations easier. However, it cannot shrink parameters to 0, thus variable selection cannot be done. $LASSO$ is a solution to this problem where variable selection is not possible. L1 penalty is a problem that is difficult to calculate and has no analytic solution, but it can shrink certain coefficients to 0. Also, something in detail will also be explained in the next subsection shown as following.

7.2 the difference between LASSO and ridge regression

It is clear that *LASSO* (Least Absolute Shrinkage and Selection Operator) is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights (the L1 term). Thus, the absolute values of weight will be generally reduced, and many will tend to be zeros. While *Ridge Regression* takes a step further and penalizes the model for the sum of squared value of the weights (the L2 term). Thus, the weights not only tend to have smaller absolute values, but also really tend to penalize the extremes of the weights, resulting in a group of weights that are more evenly distributed. In other words, *Ridge Regression* leads to both low variance (as some coefficient leads to negligible effect on prediction) and low bias (minimization of coefficient reduce the dependency of prediction on a particular variable). The difference between *Ridge* and *LASSO* regression is that *LASSO* tends to make coefficients to absolute zero as compared to *Ridge* which never sets the value of coefficient to absolute zero. By the way, *Ridge Regression* is computationally less intensive than *LASSO*.