Machine Learning MSc Coursework

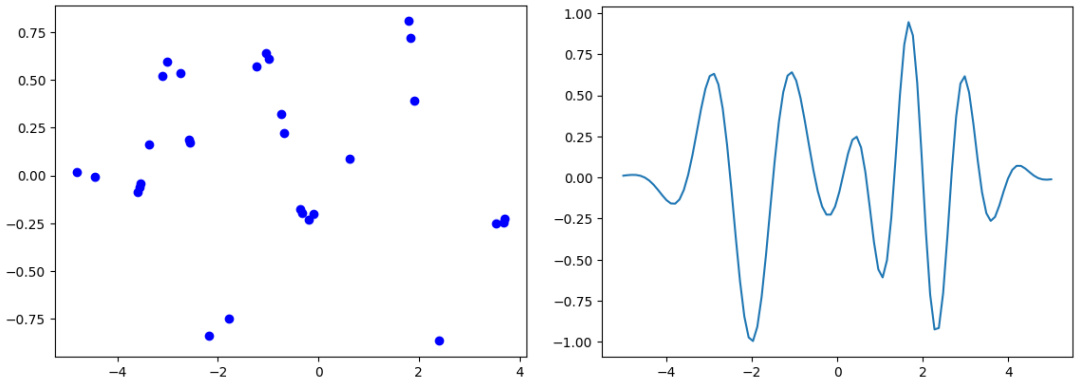
Yang Tan, yt9u22@soton.ac.uk, 34006702

December 5, 2022

1 Regression

1.1 Sample

In order to make a polynomial regression, samples should previously be produced by the required function $f(x)$. Meanwhile, x_n is randomly drawn from $-5 \leq x_n \leq 5$ with simply uniform distribution. There are 30 sample points (Fig. 1).



(a) 30 sample points (b) $f(x)$

Figure 1: Produce Samples

1.2 Model - Polynomial Regression

No mater how X transforms to a new form $\Phi(X)$, the last layer of regression is linear regression $\Phi(X) \cdot \boldsymbol{w}$. Linear combination is the key of regression and it has many elegant attributes especially about the optimization methods. Polynomial form is $\Phi_i(X) = X^i (i \geq 2)$, so polynomial regression is actually the linear combination of X^i:

$$\hat{\boldsymbol{y}} = \Phi(X) \cdot \boldsymbol{w} = \sum_{i=1}^{n} X^i \cdot w_i \quad (n \geq 1)$$

1.3 Optimization - Normal Equation

As mentioned above, polynomial regression is linear regression of X^i. Hence, its optimization is the same with linear regression. Both normal equation and gradient descent can work here. However, compared with gradient descent normal equation is optimal solution and thus it was used here.
To make residual $\boldsymbol{r} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ least, $\boldsymbol{r} \perp \hat{\boldsymbol{y}} = \Phi(X) \cdot \boldsymbol{w}$ which means linear combination of $\Phi_i(X)$ (the i-th column of $\Phi(X)$). So $\boldsymbol{r} \perp \Phi_i(X)$ and thus $\Phi^T(X) \cdot (\boldsymbol{y} - \Phi(X) \cdot \boldsymbol{w}) = 0$. Finally

$$\boldsymbol{w} = (\Phi^T(X)\Phi(X))^{-1}\Phi^T(X)\boldsymbol{y}$$

1.4 Performance vs. Polynomial Order

As indicated in Fig. 2 that train and test samples are generated in the same time and randomly selected with the ratio 30 vs 30. The regression could be normal when degree is from 1 to 5, but when degree continued to be higher the test losses are not acceptable.

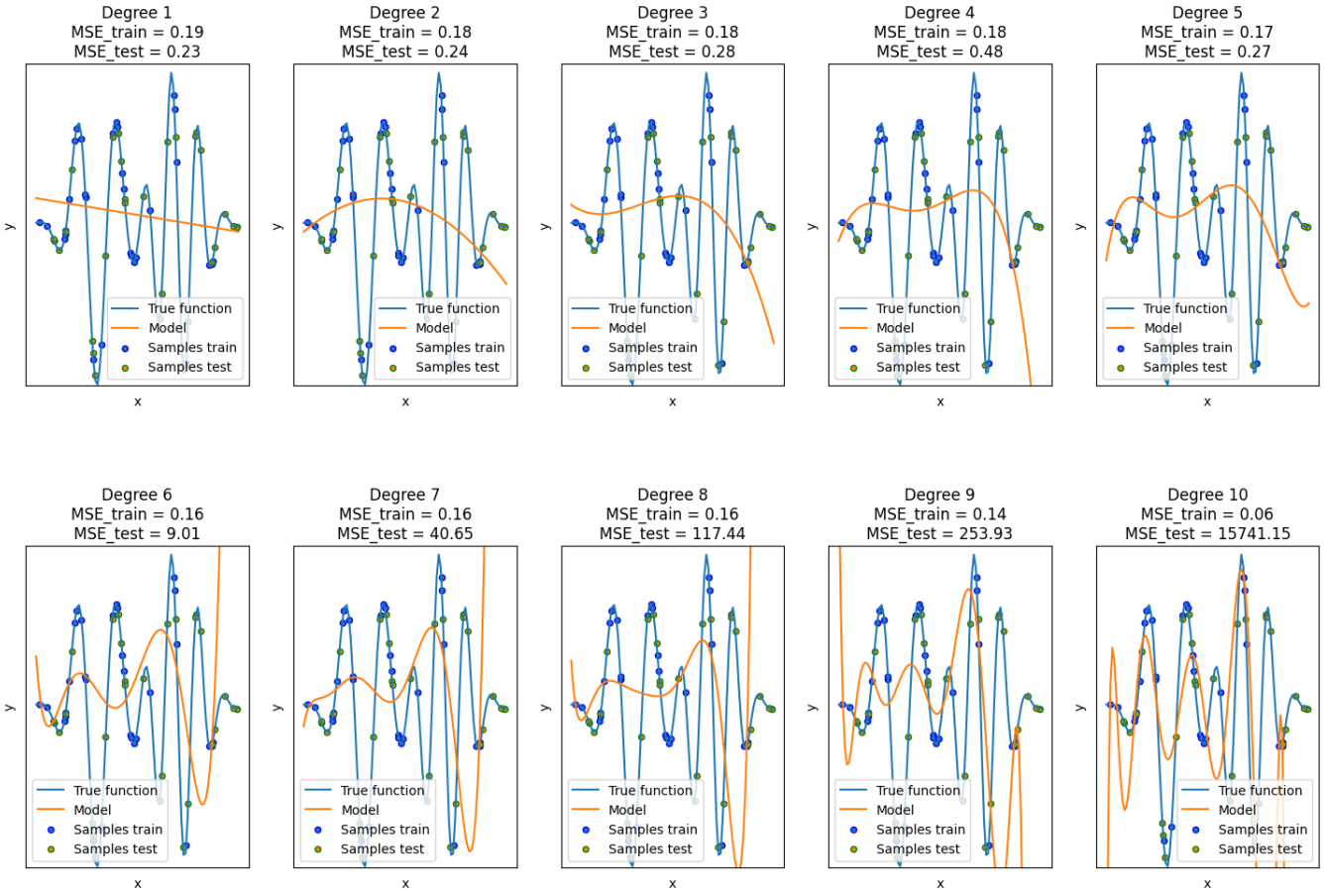


Figure 2: Produce Samples

1.5 Evaluation - Test Set , MSE loss

Image Fig. 3 presents that as the degree i grows overfitting is happening that test losses are much higher than train losses. More generally, as complexity of model increases, in the beginning underfitting may happen but after a best situation, test error will be much bigger than train error which means overfitting appears.

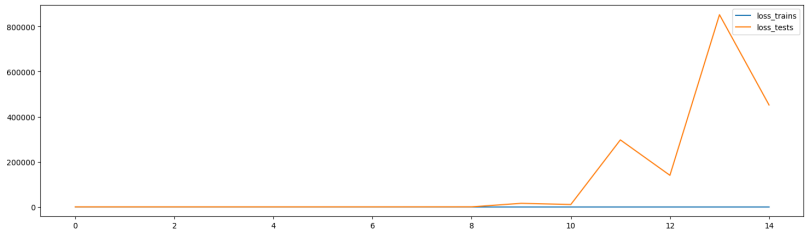


Figure 3: Mean squared error (MSE)

2 Classification

2.1 Model - Multi Logistic Regression

When it comes to classification, the objective is changed to divide continues values into discrete ones. In addition, the hypothesis $h(x)$ could be naturally seen as probability prediction. Therefore, rather than adjusting x, logistic regression model adds a active function $\delta(x)$ out of linear regression. It is obvious that $h(x)$ is not linear any more. Moreover, from probability perspective $h(x)$ is the likelihood of a class, and from information theory perspective $h(x)$ is a stochastic variable which could be used to calculate similarity with true y by cross entropy. Hence, in multi classification situation, softmax function could be used as active function to hypothesise $h(x)$ and cross entropy evaluate loss of $h(x)$:

$$\hat{Y_{N\times C}} = Softmax(X_{N\times P} \cdot W_{P\times C})$$

$$h_k(X) = \hat{Y_k} = \frac{e^{XW_{\cdot k}}}{\sum_{i=1}^{C} e^{XW_{\cdot i}}} \tag{1}$$

$$L(W) = -\sum_{n=1}^{N}\sum_{c=1}^{C} y_{nc} \ln \hat{y_{nc}} \tag{2}$$

2.2 Optimization - Gradient Descent

Because $h(x)$ is not linear any more, normal equation is unfit for classification. Referring to Taylor series, more general methods are able to solve it through approximating $L(W)$ to find a faster direction going down. Furthermore, there are first-order and second-order optimization technologies. Gradient descent belongs to first-order optimization to descend in the fastest orientation.

$$W^{(t+1)} = W^{(t)} - \eta \nabla_W L(W^{(t)}) \tag{3}$$

$$\nabla_W L(W) = \sum_{n=1}^{N} \nabla_W l_n(W)$$

Given that it is essential to be familiar with all of this to realize programs, deriving whole solution of $\nabla_W L(W)$ can help with using much simple matrix product to update parameter W in codes. Here are the processes: For one sample (x_n, y_n)

$$l_n(W) = -\sum_{c=1}^{C} y_{nc} \ln \hat{y_{nc}}$$

$$\hat{y_{nc}} = e^{a_c} / \sum_{i=1}^{C} e^{a_i} \quad a_j = x_n W_{\cdot j} = \sum_{i=1}^{P} x_{ni} \cdot w_{ij}$$

$$\frac{\partial l_n(W)}{\partial w_{ij}} = \sum_{s=1}^{C} \frac{\partial l_n}{\partial \hat{y_{ns}}} \frac{\partial \hat{y_{ns}}}{\partial w_{ij}} = \sum_{s=1}^{C}\sum_{r=1}^{C} \frac{\partial l_n}{\partial \hat{y_{ns}}} \frac{\partial \hat{y_{ns}}}{\partial a_r} \frac{\partial a_r}{\partial w_{ij}}$$

$$\frac{\partial l_n}{\partial \hat{y_{ns}}} = -\frac{y_{ns}}{\hat{y_{ns}}} \qquad \frac{\partial a_r}{\partial w_{ij}} = \delta_{rj} x_{ni} \qquad \frac{\partial \hat{y_{ns}}}{\partial a_r} = \frac{\delta_{sr} e^{a_s}}{\sum_{i=1}^{C} e^{a_i}} - \frac{e^{a_r} e^{a_s}}{(\sum_{i=1}^{C} e^{a_i})^2} = \hat{y_{ns}}(\delta_{sr} - \hat{y_{nr}})$$

$$\frac{\partial l_n(W)}{\partial w_{ij}} = \sum_{s}\sum_{r} - y_{ns}\delta_{rj} x_{ni}(\delta_{sr} - \hat{y_{nr}}) = \sum_{s} - y_{ns} x_{ni}(\delta_{sj} - \hat{y_{nj}}) = -y_{nj} x_{ni} + x_{ni}\hat{y_{nj}}$$

$$\nabla_{w_{ij}} l_n(W) = x_{ni}(\hat{y_{nj}} - y_{nj}) \tag{4}$$

$$\nabla_W L(W) = X^T_{P\times N} \cdot (\hat{Y_{N\times C}} - Y_{N\times C}) = \nabla_{P\times C} \tag{5}$$

It is significant that transforming (4) to (5) ensures clear and efficient codes without any for loops. Note also that other functions were done with only matrix operations referring to (1) and (2).

2.3 Evaluation - Test Set , Cross entropy loss

As shown in (3), η decides the rate to do gradient descend. Image Fig. 4 and Fig. 4 illustrate that if η was set big , a zig-zag happened and failed to find a local optimal. Hence it is significant to adjust the hyperparameter η in gradient descend optimization algorithms. In addition, as the round number went up, the loss decreased in Fig. 4 and finally reached a state with much less loss which produced w to predict the test set. Also, the loss of test set was close with train set which means neither overfitting nor underfitting happened.

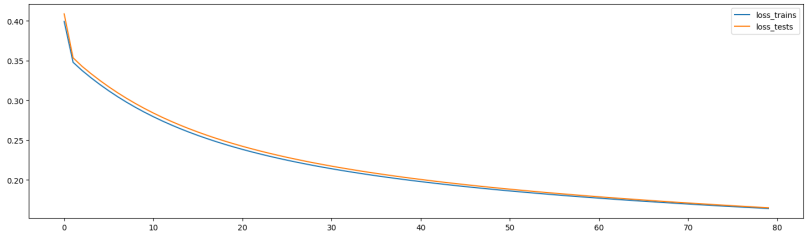


Figure 4: Cross entropy loss varies with $\eta = 0.05$

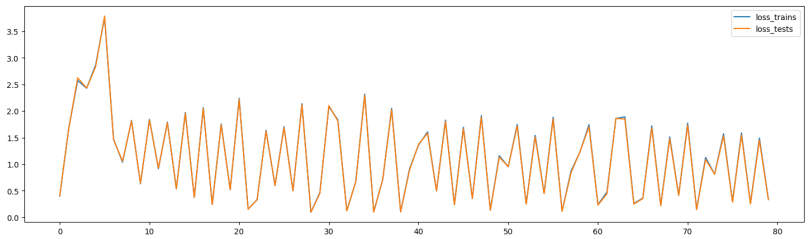


Figure 5: Cross entropy loss varies with $\eta = 0.5$

2.4 Bad case thinking

There is some thinking about (4) and (3) in bad situation. If class A is always predicted wrong, $\hat{y_{nj}} - y_{nj}$ must have a much bigger negative real number in the corresponding position which finally add a big number to W_A of class A. As a result, W_A becomes much bigger and eventually the model is more likely to output class A. That is to say, model is unable to learn class A well but prefer to select A as a result. This may be useful that if a model is more likely to predict the outcome as a single class, that means these is a problem with the model to learn this class.

2.5 Limitation

Classification models like FLD, SVM, Bayes Decision and LR all aim at finding a linear hyperplane to separate data sets. However they are unable to separate sophisticated ones such as XOR. Hence, making more than one decision boundary may be a method to solve it, for example multi layer networks are widely used to do this with better precision.

2.6 PCA

Assume that an n dimensions linear subspace has important information which can be represented by some ordered lower dimensions vectors. Projections to these vectors are principal components and singular value decomposition (SVD) is used here to find these vectors. SVD is $A = U\Sigma V^T$ and covariance matrix of sample X is $Cov(X) = \frac{1}{N-1}\widetilde{X}\widetilde{X}^T$. In addition $\widetilde{X} = X - X_{mean} = U\Sigma V^T$ and then $(N-1)Cov(X) = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T$. So SVD of $\widetilde{X}$ equals eigen-decomposition of the covariance matrix of X. Through calculating SVD or eigen-decomposition, U and Σ can be obtained and the singular vectors are columns of U and ordered by σ of Σ.
Therefore PCA was tried here to do some analysis. When using 3 principle components, there were much close losses (0.068 vs 0.067), however, when it comes to 2 , there were bigger losses (0.124 vs 0.067). This experiment showed that there are redundant information in the samples and PCA can make it better.