Project on cifar-10 Classification

Student ID: 017032910027

Name: Wang Xuhong(汪旭鸿)

Email: wangxuhongcn@163.com

Abstract

In this *Statistical Learning and Inference* course, student have a meaningful class project, which focus on the classification problem of cifar-10 dataset. Our mission is to apply three models in the course to a pattern recognition problem, and evaluate models by model selection, model inference and error analysis. To estimate the performance of different models and compare the accuracy rate to choose the best one.

Summarizing the whole work in this class report, I use training set, validation set and test set for model selection and assessment. That is, training set is to train the model, validation set is to select the model complexity, and test set is to assess the model prediction ability.

After doing all the above work with plotting cross-validation curve and learning curve , I successfully optimized the KNN, SVC and CNN models to the almost best condition.

Finally, for each of the three different models, I separately obtained the comparatively good prediction score in the cifar-10 image classification problem. The accuracy of each models is that, K-NN Classification (33.8246%), Support Vector Classification (34.2667%), Convolution Neural Networks (80.0475%).

Through this course, I learned a lot of knowledge about statistical machine learning. Including a series of standard processes for machine learning, three different machine learning models and their mathematical derivations, applications about cross-validation curves and learning curve, as well as guidelines for inferring whether the model has high bias and high variance.

Thanks for lecturer Liqing Zhang, and for teaching assistant Zhangxuan Gu!

Contents

1 Introduction

To apply three models in the course to a pattern recognition problem, and evaluate models by model selection, model inference, error analysis and choose the best one and explain your selection criterion.

To visualize some examples from the Cifar-10 dataset, as shows in Figure 1 I show a few examples of training images from each class.

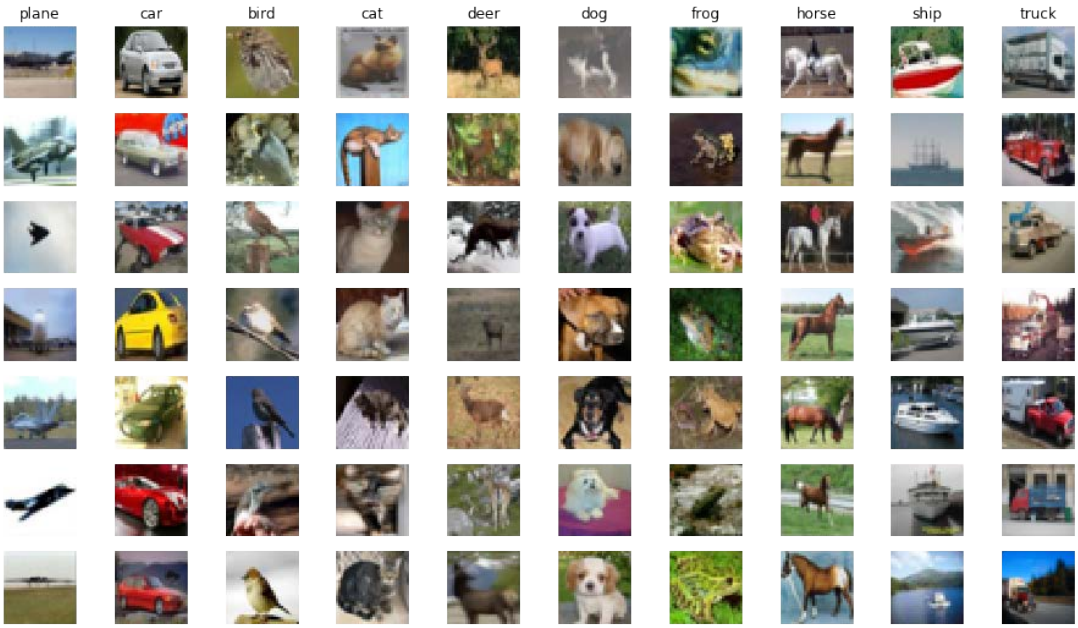

Figure 1 Cifar-10 dataset

In next part, I'll review three famous Image recognition models.

They are:

- K-Nearest Neighbors Classification(KNN)
- Support Vector Classification(SVC)
- Convolution Neural Networks (LeNet)

2 Methods

2.1 K-Nearest Neighbors Classification

K-nearest neighbor (k-NN[1]) is a basic classification and regression method. The k-nearest neighbor input is the feature vector of the instance corresponding to the point of the feature space; the output is the class of the instance, which can take multiple classes. When classifying the new instances, the algorithm predicts the majority of training examples based on the k nearest neighbors. The k-nearest neighbor method actually uses the training data set to partition the eigenvector space as a "model" of its classification. The choice of k value, distance measure and classification decision rule are the three basic elements of k-nearest neighbor method. The k-nearest neighbor method was proposed by Cover and Hart in 1968.

2.1.1 Algorithm

Input:

Training Data: $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

$x_i \in \boldsymbol{x} \subseteq R^n$ is the feature vector of the instantiation.

$y_i \in \boldsymbol{y} = \{c_1, c_2, \dots, c_K\}$ is the classification of the instantiation.

Output:

The classification which instantiation x belongs to.

Procedure:

(1) Find k nearest points to x in training dataset T according to given distance measure, and call the neighborhood of x covering these k points $N_k(x)$.

(2) Determines the category of x according to the classification rules (such as majority voting) y as formula 1.

$$y = \underset{c_j}{\operatorname{argmax}} \sum_{x_i \in N_k(x)} I(y_i = c_j) \tag{1}$$

$$I(y_i = c_j) = \begin{cases} 1, & y_i = c_j \\ 0, & y_i \neq c_j \end{cases}$$

$$i = 1,2, \dots, N; \quad j = 1,2, \dots, K$$

2.1.2 Loss Function: Euclidean distance

Considering Feature space$\chi \in R^n$, $x_{test}, x_{train} \in \chi$,

$$x_{test} = \left(x_{test}^{(1)}, x_{test}^{(2)}, \dots, x_{test}^{(n)}\right)^T$$

$$x_{train} = \left(x_{train}^{(1)}, x_{train}^{(2)}, \dots, x_{train}^{(n)}\right)^T$$

$$L_2(x_{test}, x_{train}) = \left(\sum_{l=1}^{n} \left(x_{test}^{(l)} - x_{train}^{(l)}\right)^2\right)^{\frac{1}{2}} \tag{2}$$

When it comes to the formula 2, the “nested for loop” method probably costs more than

a few hundred times than vectored method. To reduce the cost of computing, the distance between two vectors can be written as followed formula 3 by unfolding Euclidean distance formula.

$$L_2(x_{test}, x_{train}) = {x_{test}}^2 - 2x_{test}x_{train} + {x_{train}}^2 \tag{3}$$

2.2 Support Vector Classification

Support vector machines[2] is to define the most widely spaced linear classifier in the feature space; support vector machines also include kernel techniques that make it a virtually non-linear classifier. The learning strategy of SVM is to maximize the interval, which can be formalized into a convex quadratic programming (convex quadratic programming) problem, which is equal to minimizing the regularized hinge loss function. The learning algorithm of SVM is an optimization algorithm for solving convex quadratic programming.

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

2.2.1 Algorithm

Input:
A training dataset in a feature space: $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
$x_i \in \boldsymbol{x} \subseteq R^n$
$y_i \in \boldsymbol{y} = \{1, -1\}, i = 1,2, \dots, N$
Output:
Constructs a hyper-plane ($wx + b = 0$) and the classification decision function.
Procedure:
The problem of learning SVM becomes convex quadratic programming problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w\phi(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1,2, \dots, N \end{aligned} \tag{4}$$

ξ_i: To solve the linear inseparable problem, the Slack Variable ξ_i is involved.
$\phi(\cdot)$: To solve the non-linear problem, kernel trick is involved.
C: $C \geq 0$, the C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

2.2.2 Loss Function

I will first develop a commonly used loss called the Multiclass Support Vector Machine (SVM) loss. for the i-th example I are given the pixels of image x_i and the label y_i that specifies the index of the correct class.

When it comes to computing the loss function, I usually use formula 5 instead of

formula 4.

$$\min_{w,b} \sum_{i=1}^{N} [1 - y_i(wx_i + b)]_+ + \lambda \|w\|^2 \tag{5}$$

I call $[\cdot]_+$ as hinge loss function, and define:

$$[z]_+ = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

As I can see, the parameter λ can be determined by the cross-validation method, also can solve the uncertainty of the parameter W.

2.3 Convolution Neural Networks (LeNet)

Convolutional Neural Networks[3] are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. Softmax) on the last (fully-connected) layer. As Figure 1 shows, ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network.

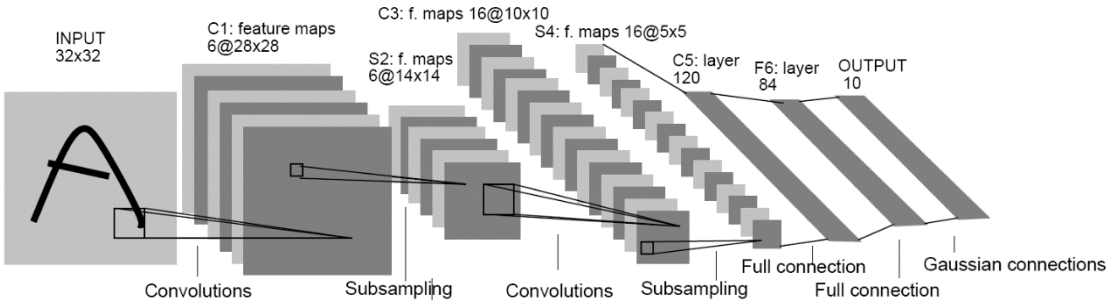

Figure 2 ConvNet architectures

2.3.1 Convolutional Layer

The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. During the forward pass, I slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As I slide the filter over the width and height of the input volume I will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. I will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. I will stack these activation maps along the depth dimension and produce the output volume.

2.3.2 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice).

2.3.3 Fully-connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. The high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication folloId by a bias offset.

2.3.4 Activation Function Layer

ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function $f(x) = max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = \tanh(x)$, and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. ReLU is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalisation accuracy.

2.3.5 CNN architectures design

As I described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. I use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. I will stack these layers to form a full ConvNet architecture.

In this case, I choose a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC].

In more detail:

- INPUT $[32 \times 32 \times 3]$ will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product betIen their Iights and a small

region they are connected to in the input volume. This may result in volume such as $[32 \times 32 \times 12]$ if I decided to use 12 filters.

- RELU layer will apply an elementwise activation function, such as the $\boldsymbol{max}(\mathbf{0}, \boldsymbol{x})$ thresholding at zero. This leaves the size of the volume unchanged ($[32 \times 32 \times 12]$).
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as $[16 \times 16 \times 12]$.
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size $[1 \times 1 \times 10]$, where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

3 Model Assessment

3.1 Assessment Methods

3.1.1 Cross-Validation Curve

Suppose I have a model with one or more unknown parameters, and a data set to which the model can be fit (the training data set). The fitting process optimizes the model parameters to make the model fit the training data as well as possible. If I then take an independent sample of validation data from the same population as the training data, it will generally turn out that the model does not fit the validation data as Ill as it fits the training data. This is called overfitting, and is particularly likely to happen when the size of the training data set is small, or when the number of parameters in the model is large. Cross-validation is a way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available.

3.1.2 Learning Curve

A learning curve shows the validation and training score of an estimator for varying numbers of training samples. It is a tool to find out how much I benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error. If both the validation score and the training score converge to a value that is too low with increasing size of the training set, I will not benefit much from more training data.

I will probably have to use an estimator or a parametrization of the current estimator that can learn more complex concepts (i.e. has a loIr bias). If the training score is much greater than the validation score for the maximum number of training samples, adding more training samples will most likely increase generalization.

3.2 K-Nearest Neighbors Classification

3.2.1 Cross-Validation Curve

Before I finish training the dataset, I must first determine the hyperparameter value, and here I use the cross-validation method. Test a small portion of the training data as a validation set to test the effect of different hyperparameter and select the best parameters for the model.

For each hyperparameter k, I calculate the correct rate for each of the 3-fold training data. I limit the range of k in [1,5,8,10,15,18,20,50].

According to the average of 3 results as **Figure** 3 and the model computational complexity, I choose **k=50** as the best parameter for the model. When I use 10000 of 50000 image to train this KNN model, the average of accuracy rate is about 28%.

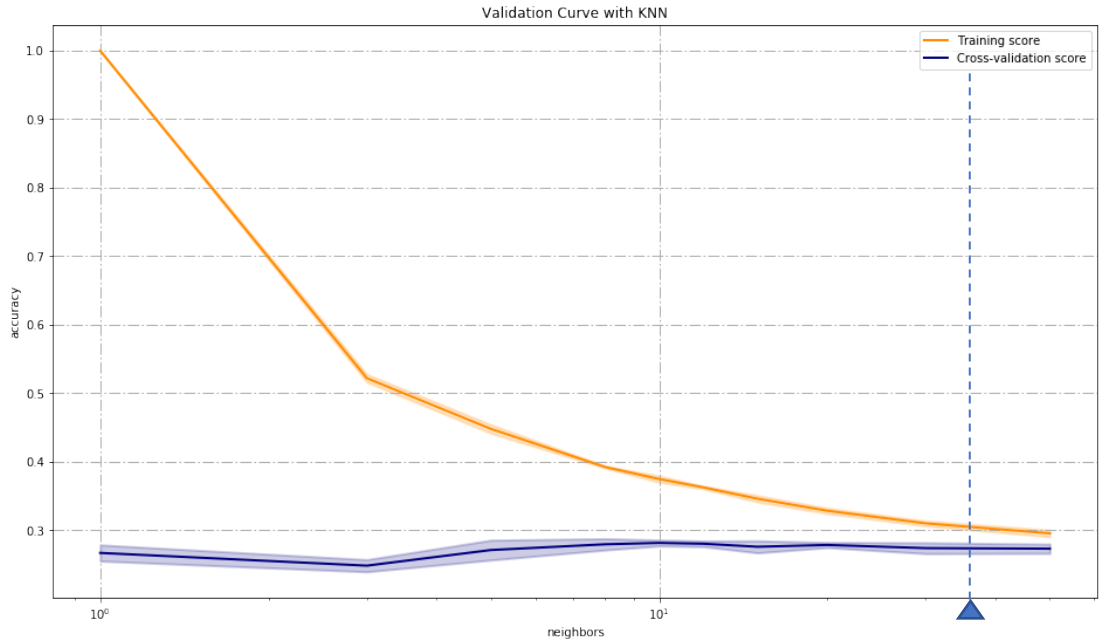

Figure 3 Cross-validation Curve of KNN

3.2.1 Learning Curve

In this case, I choose k=50 and plot the Learning Curve as **Figure** 4, it shows that: With the increasing number of training samples, the scores of training set and test set steadily increase while maintaining the relatively stable difference. It tells us that when the hyperparameter k take 50, the KNN model will benefit in more training data and finally the result of classification will be acceptable.

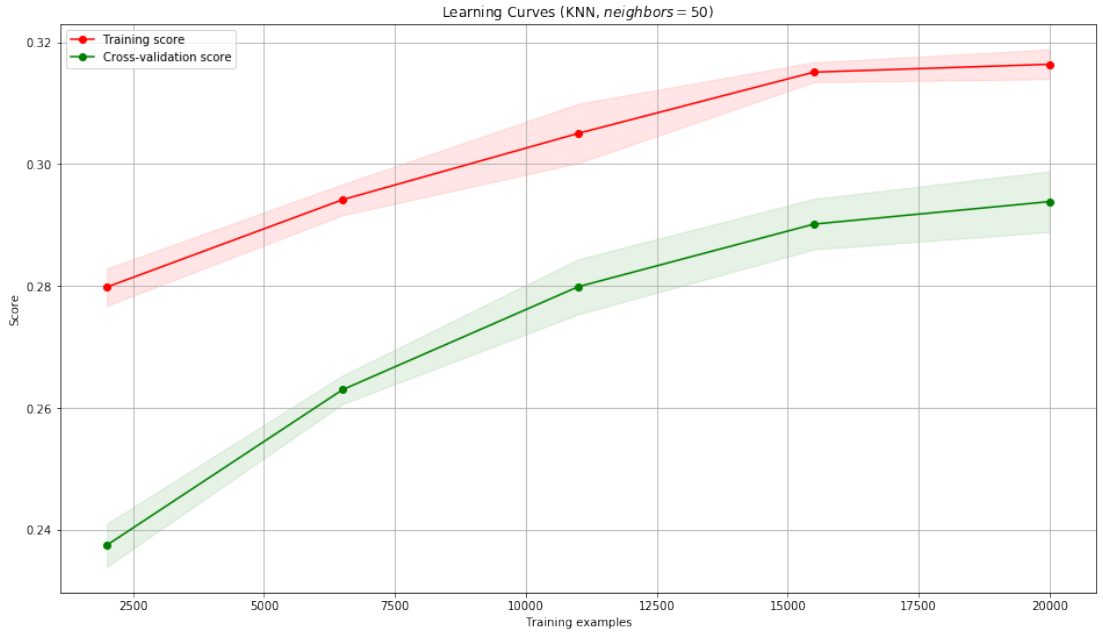

Figure 4 Learning Curve of KNN

In a word, according to the **Figure** 3 and **Figure** 4, when it comes to the KNN model of cifar-10 classification problem, the best hyperparameter k is 50. After take hyperparameter k equal 50, I make a predict in the whole dataset, the accuracy rate is about

33.8%.

3.3 Support Vector Classification

Cross-validation can make a best selection for the suitable model and helping us to choose a better hyperparameters. The validation set is used to get the classification accuracy to compare the generalization ability and then select the optimal parameters. Unluckily, limited by weak computing resources, it is not a wise choice that doing hyperparameter selection by cross-validated.

Thus, I only use learning curve combined with the previous transfer experience to make the hyperparameters select better. For each iteration, I calculate the correct rate for each of the 3-fold training data. The learning curve of Support Vector Classification is shown by **Figure 5**

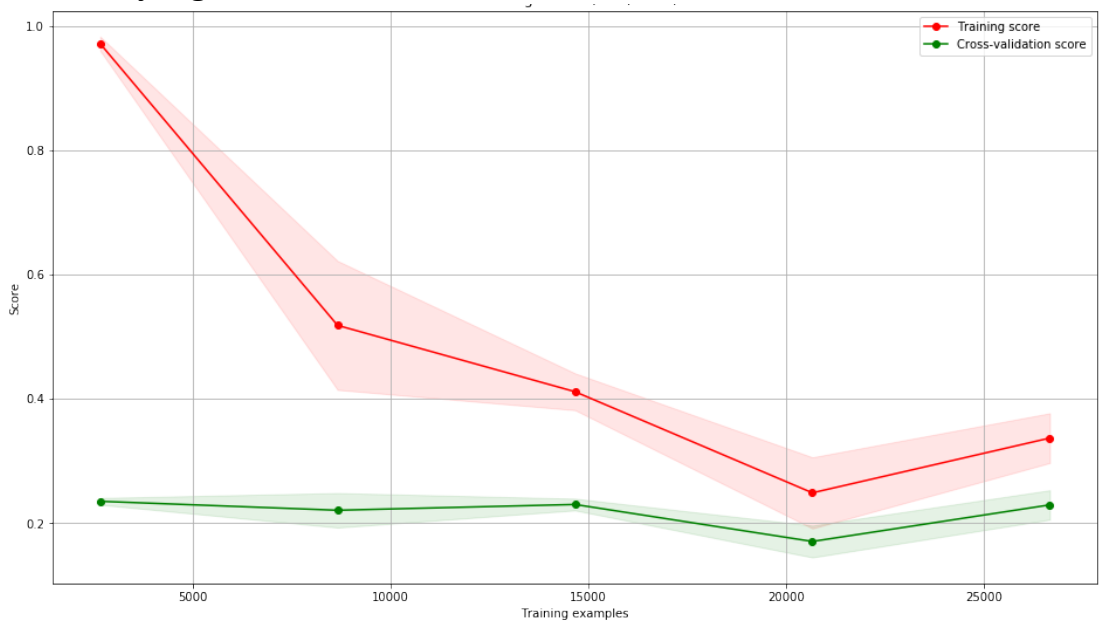

Figure 5 learning curve of Support Vector Classification

As I can see, the learning curve of Support Vector Classification shows that the model I training now seems to be a good model, which means the hyperparameters I choose are acceptable. The reason is that, with the number of training samples continues to increase, the gap betIen training set score and cross-validation set score is decreasing, and when the number of training samples reaches more than 20000, training set score and cross-validation set score at the same time steadily increased.

After all, I determining the hyperparameters for the best model, I evaluate the results on the test data. The accuracy of the test data on Cifar-10 dataset is about 34.2%.

3.3Convolution Neural Networks (LeNet)

In general, cross-validation is a very common model evaluation method in the field of machine learning. HoIver, in the field of deep learning, cross-validation is not the best method to do model evaluation, because of the particularity of the deep neural network model. In this case, I only draw the learning curve to evaluate and show the performance of a neural network model. The acc-loss learning curve of CNN is shown as **Figure 6**.

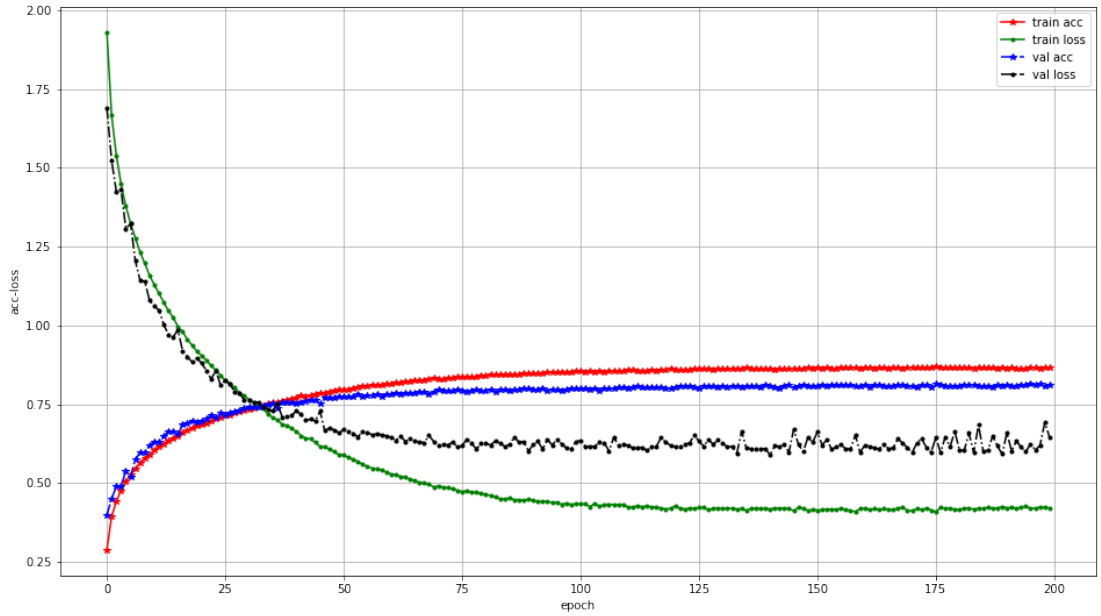

Figure 6 Acc-Loss Learning Curve of LeNet

From the **Figure 6** I know that:

- In the first 70 epochs, the score of the training set and test set continuously increases, but the highest score is about 80%.
- In the 70~200 epochs, the score of the training set and test set no longer changed basically.
- In the whole epochs, the difference betIen the training set score and the test set score remain close.
- In the whole epochs, the trends of loss and score are the similar basically.

In short, the CNN model benefits from the number of iterations. Finally, with the training of 200 epochs, the CNN classification takes about 80% accuracy in cifar-10 test dataset. This is an exciting result.

4 Summary

After doing all the above work (e.g. cross-validation and learning curve), I successfully optimized the KNN, SVC and CNN models to the almost best condition.

The final accuracy is shown as **Table 1**.

Table 1 comparison of three methods on test set

Method	Accuracy
KNN	33.8246%
SVC	34.2667%
CNN	80.0475%

The computer environment of this experiment is as **Table 2**:

Table 2 computer environment

Software	Version	Hardware	
Python	3.60	CPU	I7-6700hq
Keras	2.1.1	GPU	GTX-1060
Scikit-learn	0.18.1	Operating System	Win10_1703
Tensorflow	1.4.0		

Summarizing the above work, I use training set, validation set and test set for model selection and assessment. That is, training set is to train the model, validation set is to select the model complexity, and test set is to assess the model prediction ability. In three models, according to the results of accuracy, the conclusion can be obvious obtained:

According to **Table 1**, the accuracy on the test set:

$$\mathbf{CNN} >> \mathbf{SVC} \approx \mathbf{KNN}$$

The best model selection is CNN model.

References

[1] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

[2] “A Tutorial on Support Vector Regression”, Alex J. Smola, Bernhard Schölkopf - Statistics and Computing archive Volume 14 Issue 3, August 2004, p. 199-222.

[3] LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision[C]//Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. IEEE, 2010: 253-256.

[4] Alex Krizhevsky and Geoffrey Hinton.Learning multiple layers of features from tiny images. 2009.