Explorative Statistical Methods for Image Classification Task

Ruisheng Cao
StuID: 118033910058
211314@sjtu.edu.cn

Abstract

This paper explores a massive statistical methods to tackle the problem of Image Classification in a community Kaggle competition. We have tried different statistical methods such as Support Vector Machine, Logistic Regression, Ridge Classifier, K-Nearest Neighbours, Linear/Quadratic Discriminant Analysis. In addition, with the popularity of deep learning and neural network, we also experiment using fully connected neural network and convolution neural network. The emphasis is placed on the summarization of major advanced classification approaches and the techniques used for improving classification accuracy.

1. Introduction

Image classification refers to the task of extracting information classes from a pixel image. The intent of the classification process is to categorize all pixels in a digital image into one of several predefined classes, or "themes". The resulting raster from image classification can be used to create thematic maps. Depending on the interaction between the analyst and the computer during classification, there are two types of classification: supervised and unsupervised. We focus on supervised classification in this paper, that is the training dataset contains both image features and class labels.

Image classification is a complex process that may be affected by many factors. From preparing high-quality image dataset, preprocessing images, extracting features(feature selection), to model design, model selection, regularization methods, performance evaluation, etc. This paper examines current practices of image classification. In addition, some important issues affecting classification performance are discussed.

2. Problem Formulation

2.1. Problem statement

Supervised Image classification Problem is a classic classification problem in machine learning. For each image, given its feature vector $X = (X_1, X_2, \cdots, X_p)$, our goal is to assign a most likely class label $Y \in G = \{Y_1, Y_2, \cdots, Y_K\}$ to the sample X, while p is the dimension of features and K is the number of different classes.

2.2. Notations

We use uppercase to represent generic aspects of a variable, subscript X_j denotes the j-th dimension or component. As for a specific observed value of X, we use lowercase variable $x_i, y_i, i = 1, 2, \cdots N$, N is the total number of samples or training set size; we use bold characters to denote variables that have N components(traverse the entire data set). For example, $N \times p$ matrix $\mathbf{X}$ represents the entire training set, a.k.a. design matrix; N-vector $\mathbf{x}_j, j = 1, 2, \cdots, p$ consists of all the observations on variable X_j and $\mathbf{y}$ is a N-vector containing all observed labels. All vectors are assumed to be column vectors. The definitions here are consistent with that in book *Elements of Statistical Learning*.

In the next section, starting from binary classification task($K = 2$), we will give an overview of all models used in online Kaggle competition, and then generalize to multiclass classification problems.

3. Our Models[1]

3.1. Support Vector Machine

Support Vector Machine(SVM) learning algorithm are are among the best (and many believe are indeed the best) "off-the-shelf" supervised learning algorithms. Without loss of generality, we use label set $Y \in \{+1, -1\}$(instead of $\{0, 1\}$) as class labels. In this way, given training set $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$, a classification rule induced is

$$G(X) = sign[X^T\beta + \beta_0]$$

We can safely assume that β is a unit vector, $||\beta|| = 1$, without affecting the results. Then $f(X) = X^T\beta + \beta_0$ gives the signed distance from a input feature vector X to the hyperplane defined by $f(X) = 0$. If the classes are separable, we can find a function $f(X)$ with $y_i f(x_i) > 0, \forall i$. Hence we are able to find the hyperplane that creates the biggest

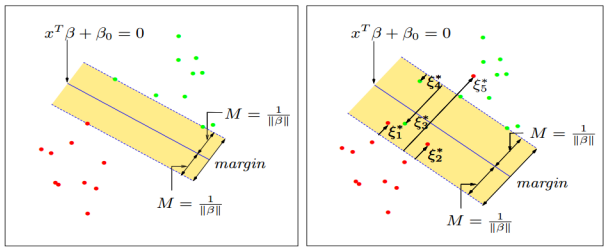

Figure 1. Support vector classifiers. The left panel shows the separable case. The right panel shows the non separable (overlap) case.

margin between training points. The optimization problem is formulated as

$$\max_{\beta,\beta_0,||\beta||=1} M$$
$$s.t. \quad y_i(x_i^T\beta+\beta_0) \geq M, i=1,2,\cdots,N$$

More conveniently, this problem can be rephrased as

$$\min_{\beta,\beta_0} \quad ||\beta||$$
$$s.t. \quad y_i(x_i^T\beta+\beta_0) \geq 1, i=1,2,\cdots,N$$

Suppose that the classes overlap in feature space, we allow for some points to be on the wrong side of the margin. Define the slack variable $\xi=(\xi_1,\xi_2,\cdots,\xi_N)$, then the general form of optimal margin classifier is defined as

$$\min_{\beta,\beta_0} \quad \frac{1}{2}||\beta||^2+C\sum_{i=1}^N \xi_i \tag{1}$$
$$s.t. \quad \xi_i>0, \quad y_i(x_i^T\beta+\beta_0)\geq 1-\xi_i, \forall i$$

To visualize the definition of margin and differences between separable and overlap cases, see Figure 1. Using Lagrange multiplier to construct Lagrange (primal) function of optimization problem (1), and changing to its dual form

$$L_D=\sum_{i=1}^N \alpha_i-\frac{1}{2}\sum_{i=1}^N\sum_{i'=1}^N \alpha_i\alpha_{i'}y_iy_{i'}x_i^Tx_{i'}$$
$$\max_{\alpha_i} \quad L_D$$
$$s.t. \quad 0\leq\alpha_i\leq C, \forall i$$
$$\sum_{i=1}^N \alpha_iy_i=0$$

We can derive a quadratic programming solution to the above dual problem.

$$\hat{\beta}=\sum_{i=1}^N \hat{\alpha}_iy_ix_i \tag{2}$$
$$\hat{f}(x)=x^T\hat{\beta}+\hat{\beta}_0$$
$$=\hat{\beta}_0+\sum_{i=1}^N \hat{\alpha}_iy_i<x,x_i> \tag{3}$$

3.1.1 Kernel Trick

Notice that in solution 3, the predicted value of $\hat{f}(x)$ is only dependent on the inner product of $<x,x_i>$. We can apply Kernel functions $K(x,x')$ instead of this inner product. Thereby, the linear separable problem is extended to linear inseparable problem given original feature space. Popular Kernel functions are

- dth-Degree polynomial: $K(x,x')=(1+<x,x'>)^d$
- Radial basis: $K(x,x')=exp(-\gamma||x-x'||^2)$
- Neural network: $K(x,x')=tanh(\kappa_1<x,x'>+\kappa_2)$

3.2. Ridge Classification

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$RSS(\beta)=(\mathbf{y}-\mathbf{X}\beta)^T(\mathbf{y}-\mathbf{X}\beta)+\lambda\beta^T\beta$$

Here, $\lambda\geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage and thus the coefficients become more robust to noise. The regression parameters are estimated using the formula

$$\hat{\beta}=(\mathbf{X}^T\mathbf{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Ridge Classifier is a classifier using Ridge regression by setting some threshold. When β^TX is larger than this threshold, the class label is 1, otherwise 0.

3.3. Logistic Regression

We now consider logistic regression. Here we are interested in binary classification, so $Y\in\{0,1\}$. Given that Y is binary valued, it therefore seems natural to choose the Bernoulli family of distributions to model the conditional distribution of Y given X. We could approach the classification problem ignoring the fact that Y is discrete-valued, and use our traditional linear regression algorithm to try to predict Y given X. Let

$$h_\theta(X)=g(\theta^TX)=\frac{1}{1+e^{-\theta^TX}}$$

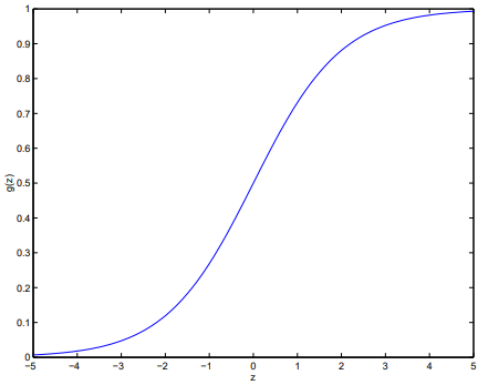

Figure 2. Shape of logistic function

where $g(z) = \frac{1}{1+e^{-z}}$ is called the *logistic function* or *sigmoid function*. Figure 2 showing $g(z)$. The decision boundary should be

$$Y = \begin{cases} 1 & \text{if } h_\theta(X) \geq 0.5 \\ 0 & \text{if } h_\theta(X) < 0.5 \end{cases}$$

Combine the two cases into one formula

$$Pr(Y|X;\theta) = h_\theta(X)^Y (1 - h_\theta(X))^{1-Y}$$

3.4. K-Nearest Neighbours

K-Nearest Neighbors algorithm (kNN) is a non-parametric method used for classification and regression. It does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the K-nearest neighbors of each data point: a query point is assigned the class label which has the most representatives within the nearest neighbors of itself. Mathematically,

$$Y = \arg\max_k \sum_{x_i \in k\text{-}Neighbour(X)} \mathbb{1} \cdot \{y_i = k\}$$

$\mathbb{1}$ is the indicator function, $k\text{-}Neighbour(X)$ is a data set representing training data points which are among the k-nearest neighbours of X.

3.5. Linear/Quadratic Discriminant Analysis

Decision theory for classification tells us that we need to know the class posteriors $Pr(Y|X)$ for optimal classification. Suppose $f_k(x)$ is the class-conditional density of X in class $Y = k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^K \pi_k = 1$, where K is the total number of classes. A simple application of Bayes theorem gives us

$$Pr(Y = k) = \frac{\pi_k f_k(X)}{\sum_{l=1}^K \pi_l f_l(X)}$$

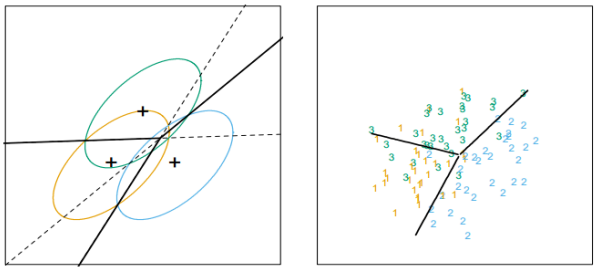

Figure 3. LDA Classification Boundaries.

Suppose that we model each class density as multivariate Gaussian

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)}$$

Linear discriminant analysis (LDA) arises in the special case when we assume that the classes have a common covariance matrix $\mathbf{\Sigma}_k = \mathbf{\Sigma}, \forall k$. If the $\mathbf{\Sigma}_k$ are not assumed to be equal, then we get *quadratic discriminant functions (QDA)*.

3.5.1 Linear Discriminant Analysis

To compare posterior probabilities of different classes, it suffices to consider the linear discriminant functions

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k \qquad (4)$$

and choose the class k with the largest $\delta_k(x)$. Mathematically, the decision rule is $Y = \arg\max_k \delta_k(x)$. In practice we do not know the parameters of the Gaussian distributions, and will need to estimate them using our training data:

- $\hat{\pi_k} = N_k/N$, where N_k is the number of class-k observations
- $\hat{\mu}_k = \sum_{y_i=k} x_i/N$
- $\hat{\mathbf{\Sigma}} = \sum_{k=1}^K \sum_{y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T/(N-K)$

Figure 3 shows the estimated decision boundaries based on a sample of size 30 each from three Gaussian distributions.

3.5.2 Quadratic Discriminant Analysis

Similar to equation 4, quadratic discriminant functions are defined as

$$\delta_k(x) = -\frac{1}{2}\log|\mathbf{\Sigma_k}| - \frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k) + \log \pi_k \qquad (5)$$

The decision boundary between each pair of classes k and l is described by a quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$.

There is also a compromise between LDA and QDA, which allows one to shrink the separate covariances of QDA toward a common covariance as in LDA. These methods are very similar in flavor to ridge regression. The regularized covariance matrices have the form

$$\hat{\mathbf{\Sigma}}_k(\alpha) = \alpha\hat{\mathbf{\Sigma}}_k + (1-\alpha)\hat{\mathbf{\Sigma}}$$

where $\hat{\mathbf{\Sigma}}$ is the pooled covariance matrix as used in LDA. Here $\alpha \in [0, 1]$ allows a continuum of models between LDA and QDA, and needs to be specified. In practice α can be chosen based on the performance of the model on validation data, or by cross validation.

3.6. From Binary to Multiclass Classification

In the above discussion, we only tackle binary classification problems, we need some strategies to apply those models in multiclass classification problems. Assume the total number of class labels is K, below are two intuitive approaches.

3.6.1 One vs Rest strategy

In One vs Rest strategy (OVR), we train one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only K classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.

3.6.2 One vs One strategy

As for One vs One strategy, we need to train one classifier per class pair. At prediction time, the class which received the most votes is selected. In this case, we require $K \times (K-1)/2$ classifiers. This method is usually slower than One vs Rest strategy, due to its O(K^2) complexity. However, this method may be advantageous for algorithms such as kernel algorithms which dont scale well with N samples. This is because each individual learning problem only involves a small subset of the data whereas, with OVR, the complete dataset is used N times.

3.7. Neural Networks

Neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function(usually non-linear) and responds with an output. Neurons are organized in layers, outputs of previous layer are the inputs of the next layer. In this way, the neurons form the entire network. The whole network has a loss function as its learning objective and uses a specific learning method such as SGD to learn the parameters.

Convolution Neural Network (CNN) also have neurons and learnable parameters. The difference is that CNNs utilize a couple of concepts called parameter sharing and local connectivity. Parameter sharing is sharing of weights by all neurons in a particular feature map. Local connectivity is the concept of each neural connected only to a subset of the input image (unlike a fully connected neural network where all the neurons are fully connected). This helps to reduce the number of parameters in the whole system and makes the computation more efficient.

Since neural network method is not the main focus in this paper, we leave out nuts and bolts here, for more details about deep learning and neural networks, see [2].

4. Experiments

In this Kaggle competition, data is collected from the famous dataset ImageNet. We select 12 classes as output labels. The dataset is split into training set and testing set. Training set contains 7800 images (650 images for each class) while test set contains 15600 images (1300 images per class). The features are 4096-dimension and are genenrated by a neural network trained on ImageNet. The task is to train a model with the features provided by the algorithms mentioned in Course X033524.

4.1. Experiment setup

In our experiment, in order to avoid repetitive work, we utilize third party python library *Scikit-learn*[3] and deep learning framework *Pytorch 0.4.1*. Our experiments are conducted on Ubuntu 16.04 LTS platform using one GPU NVIDIA GeForce GTX 970. All the implementations are available at `https://git@github.com:RhythmCao/KaggleStatLearning.git` and are completely reproducible.

4.2. Data Processing

Although the features have been provided, we need to normalize the data first. There is an issue when the features are on drastically different scales. The goal of normalization is to make every data point have the same scale so each feature is equally important. In our experiment, we try two normalization methods:

4.2.1 Min-Max

Min-max normalization is one of the most common ways to normalize data. For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a float between 0 and 1. The calculation equa-

tion for each dimension X_i is

$$X_i' = \frac{X_i - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i}$$

Min-Max algorithm guarantees all features will have the exact same scale but does not handle outliers well.

4.2.2 Z-Score

The formula for Z-score normalization is below:

$$X_i' = \frac{X_i - \bar{\mu}_i}{\sigma_i}$$

Here, μ_i is the mean value of the feature X_i and σ_i is the standard deviation of this feature. If a value is exactly equal to the mean of all the values of the feature, it will be normalized to 0. If it is below the mean, it will be a negative number, and if it is above the mean it will be a positive number. The size of those negative and positive numbers is determined by the standard deviation of the original feature. If the unnormalized data had a large standard deviation, the normalized values will be closer to 0.

Z-Score handles outliers, but does not produce features with the exact same scale.

4.3. Model selection

Once the model is chosen, in order to select the best hyper parameters, we use traditional cross validation in our experiment.

4.3.1 Cross Validation

Cross-validation is a statistical method used to evaluate the performance of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in excellent estimates that generally have a lower bias than other methods.

The procedure has a single parameter K that denotes the partitions we split the entire training set into. The corresponding method is called K-fold cross validation. If $K = N$, number of training samples, the method is called least-one-out cross validation. In our experiment, we set $K = 5$.

The general procedure is as follows:

1. Shuffle the training data set randomly
2. Divide the data set into K splits
3. Fix the hyper parameters in the model
4. For each unique split $i \in \{1, 2, \cdots, K\}$, take it as a hold out or test data set and use the remaining splits as a training data set. Train a model on the remaining $(K - 1)$ splits and evaluate it on the chosen split i. Retain the evaluation score.
5. Average the scores in the last step and choose the highest one, select the corresponding hyper parameter to train a model using the entire data set.

4.4. Ensemble Methods

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. In general, ensemble methods can be classified into two groups: sequential and paraller methods, AdaBoost and Random Forest algorithm as a representative respectively.

In our experiment, we use an averaging methods(parallel). The driving principle is to build several estimators independently and then to ensemble their predictions based on the principle of majority vote. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model.

5. Results and Analysis

In this section, we give a detailed analysis on the experiment results using models described in Section 3.

5.1. 5-fold Cross Validation Results

Take Ridge Classification as an example, the only hyper parameter is α, larger values specify stronger regularization. The cross validation accuracy and test accuracy for different settings of α are shown in Table 5.1 and Figure 4.

Table 1. CV Acc changes with α in Ridge Classification

$\log \alpha$	0	1	2	3	4	5
CV acc	0.9728	0.9765	0.9826	0.9858	0.9847	0.9826

From these illustrations, we can discover that CV accuracy increases with the increase of α at first, but will start to drop when α gets much larger. The best choice for α is around 1000.

5.2. Results of Different Models

In this subsection, we list and compare results on different models, see Table 2. Among all the methods, we may find that Logistic Regression performs best among all the models. Logistic Regression, Ridge Regression, SVM and

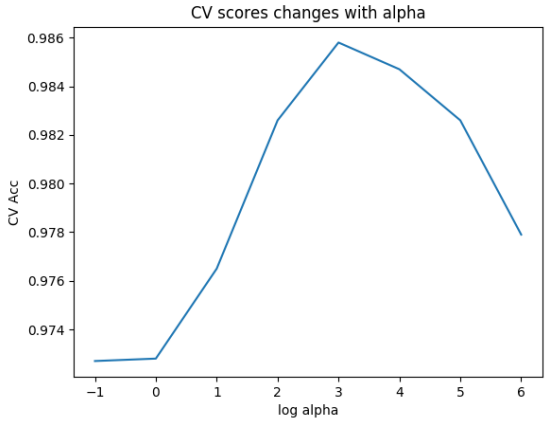

Figure 4. Cross Validation on Alpha in Ridge Classification

Methods	Ridge	Logistic	SVM	LDA	QDA	kNN
Test Acc	0.92435	0.92841	91923	0.91538	0.59914	89.44

Table 2. Accuracy on Different Statistical Models

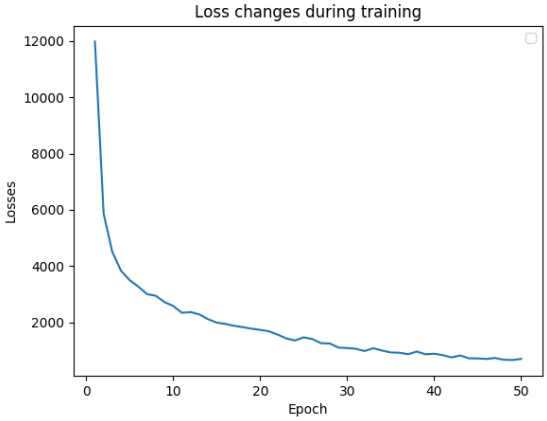

Figure 5. Loss changes during training(CNN)

LDA are more suitable in this classification problem than QDA and kNN.

5.3. Neural Network Results

We have experimented using both fully connected neural network models and convolution neural networks. Epoch loss changes during training time is illustrated in Figure 5 (CNN as an example, $4096\text{-}dim = 64\text{-}height \times 64\text{-}width$). Accuracy on validation dataset and testset is shown in Table 3.

Table 3. Accuracy using Neural Networks

Model	Best Dev Acc	Test Accuracy
Fully Connected	98.5897	0.91388
CNN	95.4487	0.82542

From Table 3, we can easily find that Fully Connected Model is more suitable than CNN Model, which seems to be a contradiction in Computer Vision field, where massive experiments have shown that CNN model is more powerful. It is because that the 4096-dim features doesn't necessarily form a 64×64 feature map in our assumption(they may be shuffled), thus parameter sharing and local connectivity do not hold any more.

5.4. Models Ensemble

In order to improve the performance of our classifier, we use majority vote ensemble method to combine the prediction results of SVM, Ridge Classification, Logistic Regression, k-NN and neural networks. Eventually, we achieve an overall improvement on test set accuracy, see Table 4.

Table 4. Final Accuracy on Testset

Method	Test Accuracy
Majority Vote	0.93247

6. Conclusion

In this paper, we analyze different methods to tackle the image classification problem, including Ridge Classification, Logistic Regression, Support Vector Machine, Linear/Quadratic Discriminant Analysis, k-Nearest Neighbours and Neural Networks.

References

[1] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[2] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.