CHAPTER

LEAST SQUARES CLASSIFICATION 26

CHAPTER CONTENTS

In Chapter 22–Chapter 25, various regression techniques for estimating real-valued output were introduced. In this chapter, the classification problem of estimating category-valued output is tackled by the LS regression method introduced in Chapter 22. In the context of classification, input $\boldsymbol{x}$ is referred to as a *pattern* and output y is referred to as a *label*.

26.1 CLASSIFICATION BY LS REGRESSION

Let us consider a binary classification problem where the label y takes either $+1$ or -1. Then the classification problem can be regarded as approximating a binary-valued function $f(\boldsymbol{x}) \in \{+1, -1\}$ (Fig. 26.1). Such a function may be naively learned by the LS method introduced in Chapter 22:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\mathrm{argmin}}\ \frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2.$$

Then, an estimate $\widehat{y}$ of the label y for a test pattern $\boldsymbol{x}$ is obtained by the sign of the learned function:

$$\widehat{y} = \begin{cases} +1 & (f_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{x}) \geq 0), \\ -1 & (f_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{x}) < 0). \end{cases} \tag{26.1}$$

Various extensions of the LS method introduced in Chapter 23, Chapter 24, and Chapter 25 may also be utilized for classification.

A MATLAB code of classification by ℓ_2-regularized LS for the Gaussian kernel model,

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n}\theta_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|^2}{2h^2}\right),$$

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00037-6

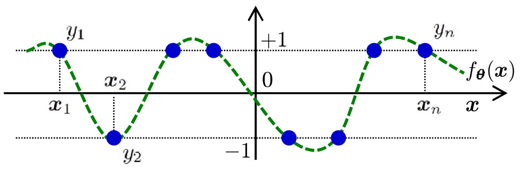

FIGURE 26.1

Binary classification as function approximation.

```
n=200; a=linspace(0,4*pi,n/2);
u=[a.*cos(a) (a+pi).*cos(a)]'+rand(n,1);
v=[a.*sin(a) (a+pi).*sin(a)]'+rand(n,1);
x=[u v]; y=[ones(1,n/2) -ones(1,n/2)]';

x2=sum(x.^2,2); hh=2*1^2; l=0.01;
k=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x')/hh);
t=(k^2+l*eye(n))\(k*y);

m=100; X=linspace(-15,15,m)'; X2=X.^2;
U=exp(-(repmat(u.^2,1,m)+repmat(X2',n,1)-2*u*X')/hh);
V=exp(-(repmat(v.^2,1,m)+repmat(X2',n,1)-2*v*X')/hh);
figure(1); clf; hold on;
contourf(X,X,sign(V'*(U.*repmat(t,1,m))));
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==-1,1),x(y==-1,2),'rx');
colormap([1 0.7 1; 0.7 1 1]); axis([-15 15 -15 15]);
```

FIGURE 26.2

MATLAB code of classification by ℓ_2-regularized LS for Gaussian kernel model.

is provided in Fig. 26.2, and its behavior is illustrated in Fig. 26.3. This shows that entangled data can be successfully classified by ℓ_2-regularized LS.

Let us consider the linear-in-input model,

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x},$$

and assign $\{+1/n_+, -1/n_-\}$ to labels instead of $\{+1, -1\}$, where n_+ and n_- denote the number of positive and negative training samples, respectively. The LS solution in this setup is given by

$$\widehat{\boldsymbol{\theta}}_{\text{LS}} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_+ - \widehat{\boldsymbol{\mu}}_-),$$

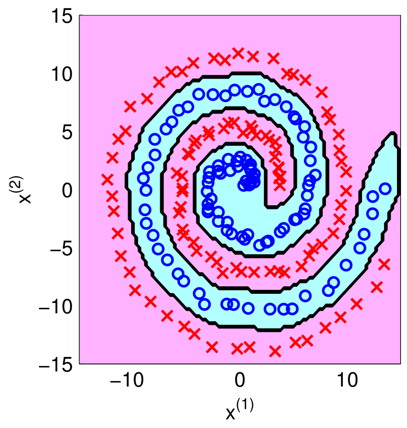

FIGURE 26.3

Example of classification by ℓ_2-regularized LS for Gaussian kernel model.

where

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top, \quad \widehat{\boldsymbol{\mu}}_+ = \frac{1}{n_+}\sum_{i:y_i=+1/n_+}^{n} \boldsymbol{x}_i, \quad \text{and} \quad \widehat{\boldsymbol{\mu}}_- = \frac{1}{n_-}\sum_{i:y_i=-1/n_-}^{n} \boldsymbol{x}_i.$$

This solution actually agrees with FDA explained in Section 12.4, which was proved to achieve the optimal classification performance when samples in the positive and negative classes follow the Gaussian distributions with a common covariance matrix. Thus, LS classification for the linear-in-input model bridges the generative and discriminative approaches.

26.2 0/1-LOSS AND MARGIN

As shown in Eq. (26.1), classification is performed based on the sign of a learned function, not the value of the function itself. Thus, in classification, the 0/1-*loss* would be more natural than the ℓ_2-loss:

$$\frac{1}{2}\Big(1 - \text{sign}\,(f_{\boldsymbol{\theta}}(\boldsymbol{x})y)\Big).$$

The 0/1-loss can be equivalently expressed as

$$\delta\Big(\text{sign}\,(f_{\boldsymbol{\theta}}(\boldsymbol{x})) \neq y\Big) = \begin{cases} 1 & (\text{sign}\,(f_{\boldsymbol{\theta}}(\boldsymbol{x})) \neq y), \\ 0 & (\text{sign}\,(f_{\boldsymbol{\theta}}(\boldsymbol{x})) = y), \end{cases}$$

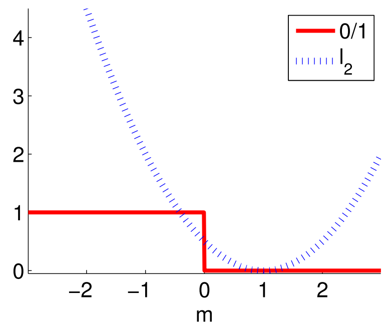

FIGURE 26.4

0/1-loss and ℓ_2-loss as functions of margin $m = f_{\boldsymbol{\theta}}(\boldsymbol{x})y$.

which means that the 0/1-loss takes 1 if $\boldsymbol{x}$ is misclassified and 0 if $\boldsymbol{x}$ is correctly classified. Thus, the 0/1-loss counts the number of misclassified samples.

The 0/1-loss is plotted as a function of $m = f_{\boldsymbol{\theta}}(\boldsymbol{x})y$ in Fig. 26.4, which takes 0 if $m > 0$ and 1 if $m \leq 0$. $m > 0$ means that $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and y have the same sign, i.e., the sample $\boldsymbol{x}$ is classified correctly. On the other hand, $m \leq 0$ corresponds to misclassification of $\boldsymbol{x}$. Thus, the value of the 0/1-loss does not depend on the magnitude of m, but only its sign. However, if m is decreased from a positive value, the 0/1-loss suddenly takes 1 if m goes below zero. Thus, having a larger m would be safer. For this reason, $m = f_{\boldsymbol{\theta}}(\boldsymbol{x})y$ is called the *margin* for sample $(\boldsymbol{x}, y)$.

As illustrated in Fig. 26.4, the 0/1-loss is a binary-valued function and the derivative is zero everywhere, except at zero where the 0/1-loss is not differentiable. For this reason, 0/1-loss minimization for a rich enough model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$,

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \Big(1 - \mathrm{sign}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)y_i\right)\Big),$$

is essentially a discrete optimization problem of assigning either $+1$ or -1 to each training sample. This is a combinatorial problem with 2^n possibilities, and the number of combinations grows exponentially with respect to n. Therefore, it cannot be solved when n is not small. Even if a solution that vanishes the 0/1-loss can be found, the solution may not be unique due to the flatness of the 0/1-loss.

In regression, the ℓ_2-loss is commonly used as training and generalization error metrics. On the other hand, in classification, although it would be natural to use the 0/1-loss as a generalization error metric since it corresponds to the misclassification rate, the 0/1-loss cannot be directly used as a training error metric due to computational intractability. For this reason, a *surrogate loss* is usually employed for training a classifier. In the LS classification method, the ℓ_2-loss is used as a surrogate for the 0/1-loss.

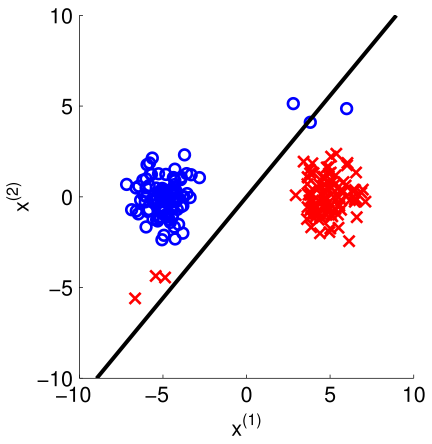

FIGURE 26.5

Example of ℓ_2-loss minimization for linear-in-input model. Since the ℓ_2-loss has a positive slope when $m > 1$, the obtained solution contains some classification error even though all samples can be correctly classified in principle.

When $y \in \{+1, -1\}$, $y^2 = 1$ and $1/y = y$ hold. Then the ℓ_2-loss, which is defined as the square of the residual $r = y - f_{\boldsymbol{\theta}}(\boldsymbol{x})$, can be expressed in terms of the margin $m = f_{\boldsymbol{\theta}}(\boldsymbol{x})y$ as

$$\begin{aligned} r^2 &= \left(y - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^2 = y^2\left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})/y\right)^2 = \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})y\right)^2 \\ &= \left(1 - m\right)^2. \end{aligned}$$

The ℓ_2-loss is plotted as a function of margin m in Fig. 26.4. The ℓ_2-loss is positive when $m < 1$, and it has a negative slope as a function of m. Therefore, differently from the 0/1-loss, the ℓ_2-loss allows gradient-based learning to reduce the error. However, the ℓ_2-loss is positive also for $m > 1$ and it has a positive slope as a function of m. Thus, when $m > 1$, ℓ_2-loss minimization reduces the margin toward $m = 1$.

Since $m = 1$ still correctly classifies the sample, reducing the margin toward $m = 1$ when $m > 1$ does not cause any problem at a glance. However, if ℓ_2-loss minimization is applied to the data illustrated in Fig. 26.5, the obtained solution contains some classification error, even though the samples can be linearly separated in principle. This is actually caused by the fact that the ℓ_2-loss has a positive slope when $m > 1$. As a surrogate for the 0/1-loss, a monotone decreasing function of margin m would be reasonable.

In Fig. 26.6, various popular surrogate loss functions are plotted, which will be explained in the following chapters.

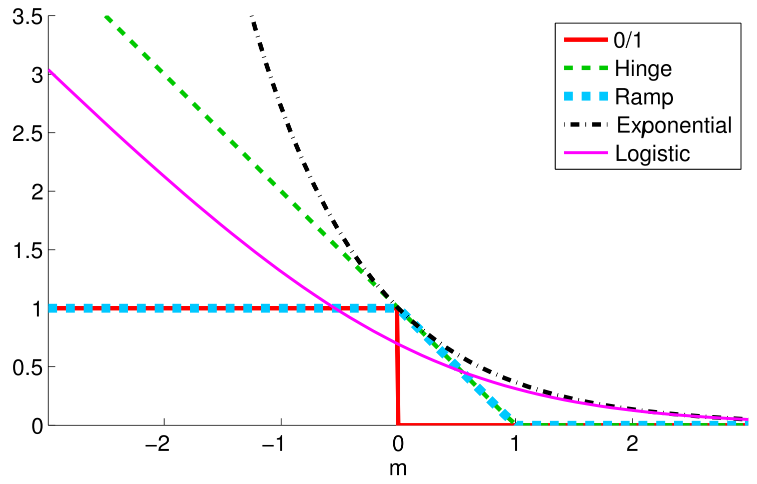

FIGURE 26.6

Popular surrogate loss functions.

26.3 MULTICLASS CLASSIFICATION

So far, binary classification was considered. However, in many practical applications of pattern recognition, the number of classes, c, may be more than two. For example, the number of classes is $c = 26$ in hand-written alphabet recognition. In this section, two methods to reduce a multiclass classification problem into a set of binary classification problems are explained. A method to directly solve the multiclass classification problems will be discussed in Chapter 27 and Chapter 28.

One way to reduce a multiclass classification problem into binary classification problems is the *one-versus-rest* method (Fig. 26.7), which considers c binary classification problems of one class versus the other classes. More specifically, for $y = 1, \ldots, c$, the yth binary classification problem assigns label $+1$ to samples in class y and -1 to samples in all other classes. Let $\widehat{f}_y(\boldsymbol{x})$ be a learned decision function for the yth binary classification problem. Then, test sample $\boldsymbol{x}$ is classified into class $\widehat{y}$ that gives the highest score:

$$\widehat{y} = \underset{y=1,\ldots,c}{\operatorname{argmax}}\, \widehat{f}_y(\boldsymbol{x}).$$

Another way to reduce a multiclass classification problem into binary classification problems is the *one-versus-one* method (Fig. 26.8), which considers $c(c-1)/2$ binary classification problems of one class versus another class. More specifically, for $y, y' = 1, \ldots, c$, the (y, y')th binary classification problem assigns label $+1$ to samples in class y and -1 to samples in class y'. Let $\widehat{f}_{y,y'}(\boldsymbol{x})$ be a learned decision function

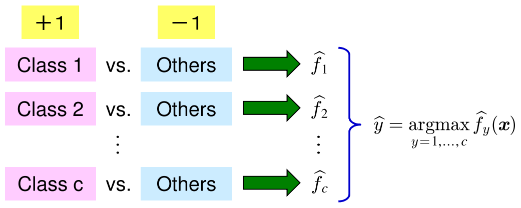

FIGURE 26.7

One-versus-rest reduction of multiclass classification problem.

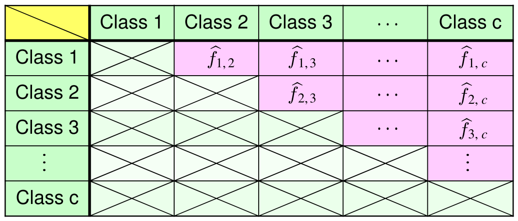

FIGURE 26.8

One-versus-one reduction of multiclass classification problem.

for the (y, y')th binary classification problem. Then, test sample $\boldsymbol{x}$ is classified into class $\widehat{y}$ that gathers the highest votes:

$$\widehat{f}_{y,y'}(\boldsymbol{x}) \geq 0 \Rightarrow \text{Vote for class } y,$$
$$\widehat{f}_{y,y'}(\boldsymbol{x}) < 0 \Rightarrow \text{Vote for class } y'.$$

One-versus-rest consists only of c binary classification problems, while one-versus-one consists of $c(c-1)/2$ binary classification problems. Thus, one-versus-rest is more compact than one-versus-one. However, a binary classification problem in one-versus-one involves samples only in two classes, while that in one-versus-rest involves samples in all classes. Thus, each binary classification problem in one-versus-one may be solved more efficiently than that in one-versus-rest. Suppose that each class contains n/c training samples, and the computational complexity of classifier training is linear with respect to the number of training samples. Then the total computational complexity for one-versus-rest and one-versus-one is both $O(cn)$. However, if classifier training takes super-linear time, which is often the case in many classification algorithms, one-versus-one is computationally more efficient than one-versus-rest.

One-versus-one would also be advantageous in that each binary classification problem is balanced. More specifically, when each class contains n/c training samples, each binary classification problem in one-versus-one contains n/c positive and n/c negative training samples, while each binary classification problem in one-versus-rest contains n/c positive and $n(c-1)/c$ negative training samples. The latter situation is often referred to as *class imbalance*, and obtaining high classification accuracy in the imbalanced case is usually more challenging than the balanced case. Furthermore, each binary classification problem in one-versus-one would be simpler than one-versus-rest, because the "rest" class in one-versus-rest usually possesses multimodality, which makes classifier training harder.

On the other hand, one-versus-one has a potential limitation that voting can be tied: for example, when $c = 3$,

$$\begin{aligned}
\widehat{f}_{1,2}(\boldsymbol{x}) \geq 0 &\Rightarrow \text{Vote for class 1,} \\
\widehat{f}_{2,3}(\boldsymbol{x}) \geq 0 &\Rightarrow \text{Vote for class 2,} \\
\widehat{f}_{1,3}(\boldsymbol{x}) < 0 &\Rightarrow \text{Vote for class 3.}
\end{aligned}$$

In such a situation, weighted voting according to the value of $\widehat{f}_{y,y'}(\boldsymbol{x})$ could be a practical option. However, it is not clear what the best way to weight is.

As discussed above, both one-versus-rest and one-versus-one approaches have pros and cons. A method to directly solve multiclass classification problems will be introduced in Chapter 27 and Chapter 28. However, the direct method does not necessarily perform better than the reduction approaches, because multiclass classification problems are usually more complicated than binary classification problems. In practice, the best approach should be selected depending on the target problem and other constraints.