CHAPTER

PROBABILISTIC CLASSIFICATION

28

CHAPTER CONTENTS

In Chapter 26 and Chapter 27, classification methods that *deterministically* assign a test pattern $\boldsymbol{x}$ into class y were discussed. In this chapter, methods of *probabilistic classification* are introduced, which choose the class probabilistically. More specifically, for a test pattern $\boldsymbol{x}$, the *class-posterior probability* $p(y|\boldsymbol{x})$ is learned in probabilistic classification. Then the class that maximizes the estimated class-posterior probability $\widehat{p}(y|\boldsymbol{x})$ is chosen:

$$\widehat{y} = \operatorname*{argmax}_{y=1,\ldots,c} \widehat{p}(y|\boldsymbol{x}).$$

The value of $\widehat{p}(y|\boldsymbol{x})$ may be interpreted as the *reliability* of assigning label y to pattern $\boldsymbol{x}$, which allows the possibility of *rejecting* the pattern $\boldsymbol{x}$ if the reliability is low.

28.1 LOGISTIC REGRESSION

In this section, a standard probabilistic classifier called *logistic regression* is introduced.

28.1.1 LOGISTIC MODEL AND MLE

The *logistic model* parameterizes the class-posterior probability $p(y|\boldsymbol{x})$ in a log-linear form as

$$q(y|\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp\left(\sum_{j=1}^{b} \theta_j^{(y)} \phi_j(\boldsymbol{x})\right)}{\sum_{y'=1}^{c} \exp\left(\sum_{j=1}^{b} \theta_j^{(y')} \phi_j(\boldsymbol{x})\right)} = \frac{\exp\left(\boldsymbol{\theta}^{(y)\top} \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{y'=1}^{c} \exp\left(\boldsymbol{\theta}^{(y')\top} \boldsymbol{\phi}(\boldsymbol{x})\right)}, \tag{28.1}$$

where the exponential function in the numerator is for restricting the output to be positive and the summation in the denominator is for restricting the output summed

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00039-X

1. Initialize $\boldsymbol{\theta}$.
2. Choose the ith training sample randomly $(\boldsymbol{x}_i, y_i)$.
3. Update parameter $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)\top}, \ldots, \boldsymbol{\theta}^{(c)\top})^\top$ to go up the gradient:

$$\boldsymbol{\theta}^{(y)} \longleftarrow \boldsymbol{\theta}^{(y)} + \varepsilon \nabla_y J_i(\boldsymbol{\theta}) \quad \text{for } y = 1, \ldots, c,$$

where $\varepsilon > 0$ is the step size, and $\nabla_y J_i$ is the gradient of $J_i(\boldsymbol{\theta}) = \log q(y_i|\boldsymbol{x}_i; \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}^{(y)}$:

$$\nabla_y J_i(\boldsymbol{\theta}) = -\frac{\exp\left(\boldsymbol{\theta}^{(y)\top}\boldsymbol{\phi}(\boldsymbol{x}_i)\right)\boldsymbol{\phi}(\boldsymbol{x}_i)}{\sum_{y'=1}^{c}\exp\left(\boldsymbol{\theta}^{(y')\top}\boldsymbol{\phi}(\boldsymbol{x}_i)\right)} + \begin{cases} \boldsymbol{\phi}(\boldsymbol{x}_i) & (y = y_i), \\ \boldsymbol{0} & (y \neq y_i). \end{cases}$$

4. Iterate 2 and 3 until convergence.

FIGURE 28.1

Stochastic gradient algorithm for logistic regression.

to one. Since the logistic model contains parameters $\{\theta_j^{(y)}\}_{j=1}^b$ for each class $y = 1, \ldots, c$, the entire parameter vector $\boldsymbol{\theta}$ is bc-dimensional:

$$\boldsymbol{\theta} = (\underbrace{\theta_1^{(1)}, \ldots, \theta_b^{(1)}}_{\text{Class 1}}, \ldots, \underbrace{\theta_1^{(c)}, \ldots, \theta_b^{(c)}}_{\text{Class } c})^\top.$$

Note that the number of parameters can be reduced from bc to $b(c-1)$ by using $\sum_{y=1}^{c} q(y|\boldsymbol{x}; \boldsymbol{\theta}) = 1$, but for simplicity the above bc-dimensional formulation is considered below.

The parameter $\boldsymbol{\theta}$ is learned by the *MLE* (Chapter 12). More specifically, $\boldsymbol{\theta}$ is determined so as to maximize the *log-likelihood*, i.e. the log-probability that the current training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are generated:

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log q(y_i|\boldsymbol{x}_i; \boldsymbol{\theta}).$$

Since the above objective function is differentiable with respect to $\boldsymbol{\theta}$, the solution may be obtained by a *stochastic gradient* algorithm (Section 15.3), as described in Fig. 28.1.

A MATLAB code of stochastic gradient ascent for logistic regression is provided in Fig. 28.2, where the log-Gaussian kernel model,

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

```
n=90; c=3; y=ones(n/c,1)*[1:c]; y=y(:);
x=randn(n/c,c)+repmat(linspace(-3,3,c),n/c,1); x=x(:);

hh=2*1^2; t=randn(n,c);
for o=1:n*1000
  i=ceil(rand*n); yi=y(i); ki=exp(-(x-x(i)).^2/hh);
  ci=exp(ki'*t); t0=t-0.1*(ki*ci)/(1+sum(ci));
  t0(:,yi)=t0(:,yi)+0.1*ki;
  if norm(t-t0)<0.000001, break, end
  t=t0;
end

N=100; X=linspace(-5,5,N)';
K=exp(-(repmat(X.^2,1,n)+repmat(x.^2',N,1)-2*X*x')/hh);
figure(1); clf; hold on; axis([-5 5 -0.3 1.8]);
C=exp(K*t); C=C./repmat(sum(C,2),1,c);
plot(X,C(:,1),'b-'); plot(X,C(:,2),'r--');
plot(X,C(:,3),'g:');
plot(x(y==1),-0.1*ones(n/c,1),'bo');
plot(x(y==2),-0.2*ones(n/c,1),'rx');
plot(x(y==3),-0.1*ones(n/c,1),'gv');
legend('q(y=1|x)','q(y=2|x)','q(y=3|x)')
```

FIGURE 28.2

MATLAB code of stochastic gradient ascent for logistic regression.

is used for approximating the class-posterior probability. The behavior of Gaussian kernel logistic regression is illustrated in Fig. 28.3, showing that the class-posterior probability is well-approximated.

The log-likelihood of logistic regression for the ith training sample $(\boldsymbol{x}_i, y_i)$ is given by

$$\log q(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = f_{y_i} - \log\left(\sum_{y=1}^{c} \exp(f_y)\right),$$

where $f_y = \boldsymbol{\theta}^{(y)\top}\boldsymbol{\phi}(\boldsymbol{x}_i)$. In numerical computation, the *log-sum-exp* term,

$$\log\left(\sum_{y=1}^{c} \exp(f_y)\right),$$

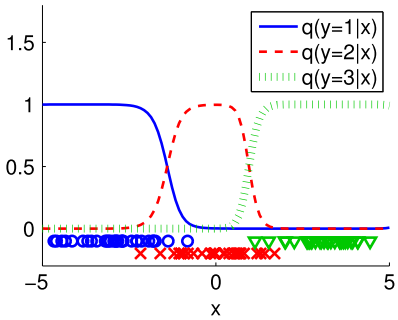

FIGURE 28.3

Example of stochastic gradient ascent for logistic regression.

is often cumbersome. When $f_y \gg 1$ for some y, $\exp(f_y)$ may cause overflow (e.g. $\exp(700) \approx 10^{300}$), resulting in $\log(\infty) = \infty$. When $f_y \ll -1$ for all $y = 1, \ldots, c$, $\exp(f_y)$ may cause underflow (e.g. $\exp(-700) \approx 10^{-300}$), resulting in $\log(0) = -\infty$. To cope with these numerical problems, the following expression of the log-sum-exp term is useful in practice:

$$\begin{aligned}\log\left(\sum_{y=1}^{c}\exp(f_y)\right) &= \log\left(\exp(f_{y'})\left(\sum_{y=1}^{c}\exp\left(f_y - f_{y'}\right)\right)\right)\\ &= f_{y'} + \log\left(\sum_{y=1}^{c}\exp\left(f_y - f_{y'}\right)\right),\end{aligned}$$

where $y' = \mathrm{argmax}_{y=1,\ldots,c}\{{\boldsymbol{\theta}^{(y)}}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\}$. With this expression, overflow can be avoided because $f_y - f_{y'} \leq 0$ for all $y = 1, \ldots, c$, and underflow does not cause $\log(0)$ because $f_y - f_{y'} = 0$ holds for $y = y'$.

28.1.2 LOSS MINIMIZATION VIEW

Let us focus on the binary classification problem for $y \in \{+1, -1\}$. Then the sum-to-one constraint,

$$q(y = +1|\boldsymbol{x}; \boldsymbol{\theta}) + q(y = -1|\boldsymbol{x}; \boldsymbol{\theta}) = 1,$$

implies that the number of parameters in the logistic model can be reduced from $2b$ to b:

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{1 + \exp\left(-y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)},$$

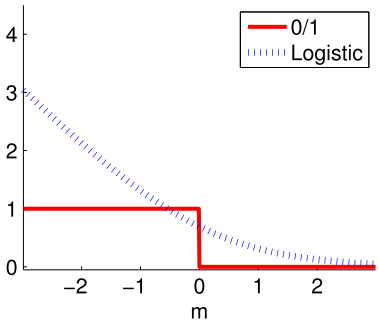

FIGURE 28.4

Logistic loss.

where $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$. Then the maximum log-likelihood criterion for this simplified model can be expressed as

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log\left(1 + \exp(-m_i)\right), \tag{28.2}$$

where $m_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) y_i$ is the *margin* for the ith training sample $(\boldsymbol{x}_i, y_i)$. Eq. (28.2) can be interpreted as *surrogate loss* minimization (see Section 26.2) with the *logistic loss* (Fig. 28.4):

$$\log(1 + \exp(-m)).$$

Thus, although logistic regression was derived in a very different framework from LS classification (Chapter 26) and support vector classification (Chapter 27), they can all be interpreted as surrogate loss minimization.

28.2 LS PROBABILISTIC CLASSIFICATION

In this section, an alternative to logistic regression based on LS fitting of the class-posterior probability is introduced.

Let us model the class-posterior probability $p(y|\boldsymbol{x})$ by the linear-in-parameter model:

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}^{(y)}) = \sum_{j=1}^{b} \theta_j^{(y)} \phi_j(\boldsymbol{x}) = {\boldsymbol{\theta}^{(y)}}^{\top} \boldsymbol{\phi}(\boldsymbol{x}). \tag{28.3}$$

Differently from logistic model (28.1), the above model for class y depends only on the parameter for class y, $\boldsymbol{\theta}^{(y)} = (\theta_1^{(y)}, \ldots, \theta_b^{(y)})^{\top}$, because the normalization term is not included.

In LS probabilistic classification, the class-posterior model $q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)})$ is learned for each class separately. More specifically, the parameter $\boldsymbol{\theta}^{(y)}$ is learned to minimize the expected squared error to the true class-posterior probability $p(y|\boldsymbol{x})$:

$$\begin{aligned} J_y(\boldsymbol{\theta}^{(y)}) &= \frac{1}{2}\int \left(q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)}) - p(y|\boldsymbol{x})\right)^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \frac{1}{2}\int q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)})p(y|\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &\quad + \frac{1}{2}\int p(y|\boldsymbol{x})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{28.4}$$

where $p(\boldsymbol{x})$ denotes the marginal probability density of training input samples $\{\boldsymbol{x}_i\}_{i=1}^n$.

The second term in Eq. (28.4), $p(y|\boldsymbol{x})p(\boldsymbol{x})$, can be expressed as

$$p(y|\boldsymbol{x})p(\boldsymbol{x}) = p(\boldsymbol{x},y) = p(\boldsymbol{x}|y)p(y),$$

where $p(\boldsymbol{x}|y)$ denotes the *class-conditional probability density* for samples in class y, $\{\boldsymbol{x}_i\}_{i:y_i=y}$, and $p(y)$ denotes the *class-prior probability* for training output samples $\{y_i\}_{i=1}^n$. The first two terms in J_y,

$$\int q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \quad \text{and} \quad \int q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)})p(y)p(\boldsymbol{x}|y)\mathrm{d}\boldsymbol{x},$$

are the expectations over $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|y)$, respectively, although they are not accessible. Here, the expectations are approximated by the sample averages:

$$\frac{1}{n}\sum_{i=1}^n q(y|\boldsymbol{x}_i;\boldsymbol{\theta}^{(y)})^2 \quad \text{and} \quad \frac{1}{n_y}\sum_{i:y_i=y} q(y|\boldsymbol{x}_i;\boldsymbol{\theta}^{(y)})p(y),$$

where n_y denotes the number of training samples in class y. Further approximating $p(y)$ by the sample ratio n_y/n, ignoring the third term in Eq. (28.4) because it is constant, and including the ℓ_2-regularizer yield the following training criterion:

$$\begin{aligned} \widehat{J}_y(\boldsymbol{\theta}^{(y)}) &= \frac{1}{2n}\sum_{i=1}^n q(y|\boldsymbol{x}_i;\boldsymbol{\theta}^{(y)})^2 - \frac{1}{n}\sum_{i:y_i=y} q(y|\boldsymbol{x}_i;\boldsymbol{\theta}^{(y)}) + \frac{\lambda}{2n}\|\boldsymbol{\theta}^{(y)}\|^2 \\ &= \frac{1}{2n}\boldsymbol{\theta}^{(y)\top}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\theta}^{(y)} - \frac{1}{n}\boldsymbol{\theta}^{(y)\top}\boldsymbol{\Phi}^\top\boldsymbol{\pi}^{(y)} + \frac{\lambda}{2n}\|\boldsymbol{\theta}^{(y)}\|^2, \end{aligned}$$

where $\boldsymbol{\pi}^{(y)} = (\pi_1^{(y)},\ldots,\pi_n^{(y)})^\top$ is defined as $\pi_i^{(y)} = \begin{cases} 1 & (y_i = y), \\ 0 & (y_i \neq y). \end{cases}$ Since $\widehat{J}_y(\boldsymbol{\theta}^{(y)})$ is a quadratic function, its minimizer can be obtained analytically by setting its derivative to zero:

$$\widehat{\boldsymbol{\theta}}^{(y)} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\pi}^{(y)}.$$

```
n=90; c=3; y=ones(n/c,1)*[1:c]; y=y(:);
x=randn(n/c,c)+repmat(linspace(-3,3,c),n/c,1); x=x(:);

hh=2*1^2; x2=x.^2; l=0.1; N=100; X=linspace(-5,5,N)';
k=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x')/hh);
K=exp(-(repmat(X.^2,1,n)+repmat(x2',N,1)-2*X*x')/hh);
for yy=1:c
  yk=(y==yy); ky=k(:,yk);
  ty=(ky'*ky+l*eye(sum(yk)))\(ky'*yk);
  Kt(:,yy)=max(0,K(:,yk)*ty);
end
ph=Kt./repmat(sum(Kt,2),1,c);

figure(1); clf; hold on; axis([-5 5 -0.3 1.8]);
plot(X,ph(:,1),'b-'); plot(X,ph(:,2),'r--');
plot(X,ph(:,3),'g:');
plot(x(y==1),-0.1*ones(n/c,1),'bo');
plot(x(y==2),-0.2*ones(n/c,1),'rx');
plot(x(y==3),-0.1*ones(n/c,1),'gv');
legend('p(y=1|x)','p(y=2|x)','p(y=3|x)')
```

FIGURE 28.5

MATLAB code for LS probabilistic classification.

However, differently from logistic model (28.1), current linear-in-parameter model (28.3) can take a negative value and may not be summed to one. To cope with this problem, the solution is postprocessed as

$$\widehat{p}(y|\boldsymbol{x}) = \frac{\max(0, \widehat{\boldsymbol{\theta}}^{(y)\top}\boldsymbol{\phi}(\boldsymbol{x}))}{\sum_{y'=1}^{c} \max(0, \widehat{\boldsymbol{\theta}}^{(y')\top}\boldsymbol{\phi}(\boldsymbol{x}))}. \tag{28.5}$$

A MATLAB code of LS probabilistic classification for the Gaussian kernel model,

$$q(y|\boldsymbol{x};\boldsymbol{\theta}^{(y)}) = \sum_{j:y_j=y} \theta_j^{(y)} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_j\|^2}{2h^2}\right),$$

is provided in Fig. 28.5, and its behavior is illustrated in Fig. 28.6. This shows that almost the same solution as logistic regression (see Fig. 28.3) can be obtained by LS probabilistic classification.

As illustrated above, LS probabilistic classification behaves similarly to logistic regression. In logistic regression, logistic model (28.1) containing bc parameters is

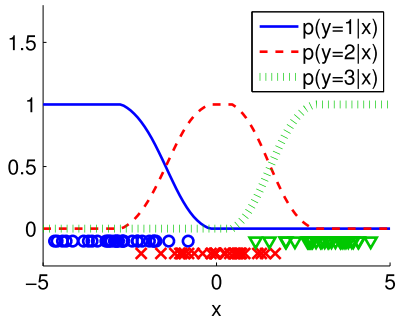

FIGURE 28.6

Example of LS probabilistic classification for the same data set as Fig. 28.3.

trained once for all classes. On the other hand, in LS probabilistic classification, linear-in-parameter model (28.3) containing only b parameters is trained for each class $y = 1, \ldots, c$. When c is large, training small linear-in-parameter models containing only b parameters c times would be computationally more efficient than training a big logistic model containing bc parameters once. Furthermore, the solution of LS probabilistic classification can be obtained analytically. However, it involves *ad hoc* postprocessing, which may corrupt the solution when the number of training samples is small. Thus, using logistic regression would be more reliable when the number of training samples is small, while using LS probabilistic classification would be computationally more efficient when the number of training samples is large.