CHAPTER

ONLINE LEARNING 31

CHAPTER CONTENTS

The supervised learning methods explained so far handled all training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ at the same time, which is called *batch learning*. On the other hand, when training samples are provided one by one in a sequential manner, it would be effective to perform *online learning*, i.e. a new training sample is incrementally learned upon the current learned result. Online learning is also useful when the number of training samples is so large that all training samples cannot be stored in memory. In this chapter, various online learning algorithms are introduced.

For simplicity, only the *linear-in-input model* is considered in this chapter:

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

However, all methods introduced in this section can be easily extended to $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x})$, *linear-in-parameter models* (see Section 21.1) with basis function $\boldsymbol{\phi}(\boldsymbol{x})$.

31.1 STOCHASTIC GRADIENT DESCENT

The *stochastic gradient* algorithm introduced in Section 22.4 is one of the typical online learning algorithms, which updates the parameter to reduce the loss for a new training sample.

More specifically, for the new training sample $(\boldsymbol{x}, y)$ and a loss function J, the parameter $\boldsymbol{\theta}$ is updated along the gradient of the loss ∇J as

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \varepsilon \nabla J(\boldsymbol{\theta}),$$

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00042-X

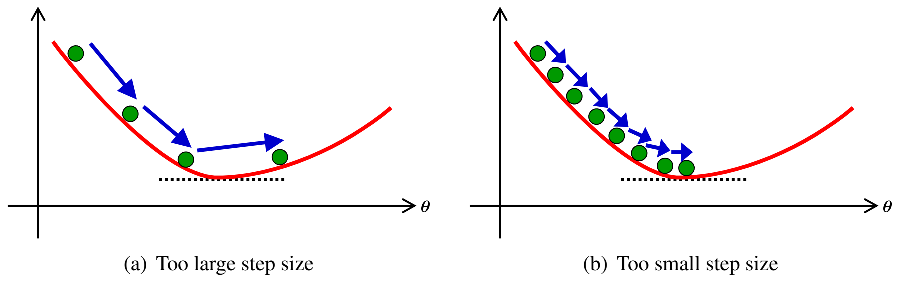

FIGURE 31.1

Choice of step size. Too large step size overshoots the optimal solution, while too small step size yields slow convergence.

where $\varepsilon > 0$ denotes the step size. See Fig. 22.6 for the detailed algorithm of the stochastic gradient method.

Although stochastic gradient is simple and easy to use, choice of step size ε is often cumbersome in practice, as illustrated in Fig. 31.1: too large step size overshoots the optimal solution, while too small step size yields slow convergence. Starting from a large ε and then reducing ε gradually, called *simulated annealing*, would be useful to control the step size. However, choice of initial ε and the decreasing factor of ε is not straightforward in practice.

For simple loss functions such as the squared loss,

$$J(\boldsymbol{\theta}) = \frac{1}{2}\left(y - \boldsymbol{\theta}^\top \boldsymbol{x}\right)^2,$$

it is actually possible to find the minimizer with respect to $\boldsymbol{\theta}$ analytically. Then step size selection no longer matters, which is highly appealing in practice. However, this often changes the parameter $\boldsymbol{\theta}$ drastically, implying that the knowledge acquired so far can be completely spoiled by addition of a single, possibly noisy training sample. Thus, in practice, it is important to choose the step size to be not too large.

31.2 PASSIVE-AGGRESSIVE LEARNING

In this section, a simple online learning method called *passive-aggressive learning* [32] is introduced, which better controls the step size.

More specifically, in passive-aggressive learning, the amount of change from the current solution $\widetilde{\boldsymbol{\theta}}$ is included as a penalty term:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \left[J(\boldsymbol{\theta}) + \frac{\lambda}{2}\|\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}\|^2 \right], \tag{31.1}$$

where $\lambda > 0$ is the *passiveness* parameter. Minimizing the first term in Eq. (31.1) corresponds to aggressively updating the parameter, while minimizing the second term in Eq. (31.1) corresponds to passively keeping the previous solution. Below, passive-aggressive algorithms for classification and regression are introduced.

31.2.1 CLASSIFICATION

For classification, let us employ the *squared hinge loss* (see Fig. 27.12):

$$J(\boldsymbol{\theta}) = \frac{1}{2}\left(\max\left\{0, 1-m\right\}\right)^2,$$

where $m = \boldsymbol{\theta}^\top \boldsymbol{x} y$ denotes the *margin*. Then the solution of passive-aggressive classification can be obtained analytically as follows.

First, the squared hinge loss can be expressed as

$$J(\boldsymbol{\theta}) = \min_{\xi} \frac{1}{2}\xi^2 \quad \text{subject to } \xi \geq 1-m \text{ and } \xi \geq 0.$$

Note that $\xi \geq 0$ is actually automatically fulfilled because the objective function only contains ξ^2 and $\xi = 0$ is the solution when $1 - m \leq 0$. Then the objective function in Eq. (31.1) can be written as

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta},\xi}\left[\frac{1}{2}\xi^2 + \frac{\lambda}{2}\|\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}\|^2\right] \quad \text{subject to } \xi \geq 1-m. \tag{31.2}$$

When $1 - m \leq 0$, $\xi = 0$ is the solution and thus $\widehat{\boldsymbol{\theta}} = \widetilde{\boldsymbol{\theta}}$ is obtained. This means that the new solution $\widehat{\boldsymbol{\theta}}$ is the same as the current solution $\widetilde{\boldsymbol{\theta}}$.

When $1 - m > 0$, let us introduce the *Lagrange multiplier* α to constrained optimization problem (31.2) and define the *Lagrange function* (see Fig. 23.5) as follows:

$$L(\boldsymbol{\theta}, \xi, \alpha) = \frac{1}{2}\xi^2 + \frac{\lambda}{2}\|\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}\|^2 + \alpha(1 - m - \xi). \tag{31.3}$$

Then the KKT conditions (Fig. 27.7) yield

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{0} \quad &\Longrightarrow \quad \boldsymbol{\theta} = \widetilde{\boldsymbol{\theta}} + \frac{\alpha y}{\lambda}\boldsymbol{x}, \\ \frac{\partial L}{\partial \xi} = 0 \quad &\Longrightarrow \quad \xi = \alpha. \end{aligned} \tag{31.4}$$

Substituting these conditions into Lagrange function (31.3) and eliminating $\boldsymbol{\theta}$ and ξ yield

$$L(\alpha) = -\frac{\alpha^2}{2}\left(\frac{\|\boldsymbol{x}\|^2}{\lambda} + 1\right) + \alpha\left(1 - \widetilde{\boldsymbol{\theta}}^\top \boldsymbol{x} y\right).$$

Taking the derivative of $L(\alpha)$ and setting it at zero give the maximizer $\widehat{\alpha}$ analytically as

$$\widehat{\alpha} = \frac{1 - \widetilde{\boldsymbol{\theta}}^\top \boldsymbol{x} y}{\|\boldsymbol{x}\|^2/\lambda + 1}.$$

```
1. Initialize $\boldsymbol{\theta} \longleftarrow \mathbf{0}$.
2. Update the parameter $\boldsymbol{\theta}$ using a new training sample $(\boldsymbol{x}, y)$ as
       $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \frac{y \max(0, 1 - \boldsymbol{\theta}^\top \boldsymbol{x} y)}{\|\boldsymbol{x}\|^2 + \lambda} \boldsymbol{x}$.
3. Go to Step 2.
```

FIGURE 31.2

Algorithm of passive-aggressive classification.

Substituting this back into Eq. (31.4) gives

$$\widehat{\boldsymbol{\theta}} = \widetilde{\boldsymbol{\theta}} + \frac{(1 - \widetilde{\boldsymbol{\theta}}^\top \boldsymbol{x} y) y}{\|\boldsymbol{x}\|^2 + \lambda} \boldsymbol{x}.$$

Altogether, the final solution $\widehat{\boldsymbol{\theta}}$ is expressed as

$$\widehat{\boldsymbol{\theta}} = \widetilde{\boldsymbol{\theta}} + \frac{y \max(0, 1 - \widetilde{\boldsymbol{\theta}}^\top \boldsymbol{x} y)}{\|\boldsymbol{x}\|^2 + \lambda} \boldsymbol{x}.$$

The algorithm of passive-aggressive classification is summarized in Fig. 31.2.

A MATLAB code for passive-aggressive classification is provided in Fig. 31.3, and its behavior is illustrated in Fig. 31.4. This shows that, after three iterations, almost the same solution as the final one can be obtained that well separates positive and negative samples. In this implementation, the intercept of the linear model is expressed by augmenting input vector $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})$ as $(\boldsymbol{x}^\top, 1)^\top$:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top (\boldsymbol{x}^\top, 1)^\top = \sum_{j=1}^{d} \theta_j x^{(j)} + \theta_{d+1}.$$

Note that, for the ordinary hinge loss (Fig. 27.12(a)),

$$J(\boldsymbol{\theta}) = \max \left\{0, 1 - m\right\},$$

the passive-aggressive algorithm still gives an analytic update formula:

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + y \min \left\{ \frac{1}{\lambda}, \frac{\max(0, 1 - m)}{\|\boldsymbol{x}\|^2} \right\} \boldsymbol{x}.$$

31.2.2 REGRESSION

The idea of passive-aggressive learning can also be applied to regression. For residual $r = y - \boldsymbol{\theta}^\top \boldsymbol{x}$, let us employ the ℓ_2-loss and the ℓ_1-loss (Fig. 25.2):

$$J(\boldsymbol{\theta}) = \frac{1}{2} r^2 \quad \text{and} \quad J(\boldsymbol{\theta}) = |r|.$$

```
n=200; x=[randn(1,n/2)-5 randn(1,n/2)+5; 5*randn(1,n)]';
y=[ones(n/2,1);-ones(n/2,1)];
x(:,3)=1; p=randperm(n); x=x(p,:); y=y(p);
t=zeros(3,1); l=1;
for i=1:n
  xi=x(i,:)'; yi=y(i);
  t=t+yi*max(0,1-t'*xi*yi)/(xi'*xi+l)*xi;
end
figure(1); clf; hold on; axis([-10 10 -10 10]);
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==-1,1),x(y==-1,2),'rx');
plot([-10 10],-(t(3)+[-10 10]*t(1))/t(2),'k-');
```

FIGURE 31.3

MATLAB code for passive-aggressive classification.

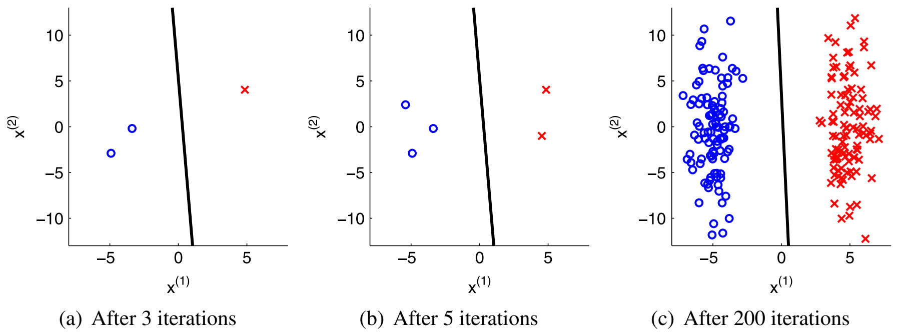

FIGURE 31.4

Example of passive-aggressive classification.

Then, with a similar derivation to the classification case, the update formulas for passive-aggressive regression can be obtained analytically as

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \frac{r}{\|\boldsymbol{x}\|^2 + \lambda}\boldsymbol{x} \quad \text{and} \quad \boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \operatorname{sign}(r) \min\left\{\frac{1}{\lambda}, \frac{|r|}{\|\boldsymbol{x}\|^2}\right\}\boldsymbol{x},$$

where sign (r) denotes the sign of r. As shown in Section 22.4, the stochastic gradient update rule for the ℓ_2-loss is given by

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \varepsilon r \boldsymbol{x},$$

```
n=50; N=1000; x=linspace(-3,3,n)'; x=x(randperm(n));
pix=pi*x; y=sin(pix)./(pix)+0.1*x+0.05*randn(n,1);
hh=2*0.3^2; t=randn(n,1); l=1;
for i=1:n
  ki=exp(-(x-x(i)).^2/hh);
  t=t+(y(i)-t'*ki)/(ki'*ki+l)*ki;
end
X=linspace(-3,3,N)';
K=exp(-(repmat(X.^2,1,n)+repmat(x.^2',N,1)-2*X*x')/hh);
F=K*t;
figure(1); clf; hold on; axis([-2.8 2.8 -0.5 1.2]);
plot(X,F,'g-'); plot(x,y,'bo');
```

FIGURE 31.5

MATLAB code for passive-aggressive regression with the ℓ_2-loss.

where $\varepsilon > 0$ is the step size. This means that passive-aggressive regression corresponds to stochastic gradient with step size $\varepsilon = 1/(\|\boldsymbol{x}\|^2 + \lambda)$, and thus the step size is adaptively chosen based on the sample $\boldsymbol{x}$.

A MATLAB code for passive-aggressive regression with the ℓ_2-loss is provided in Fig. 31.5.

31.3 ADAPTIVE REGULARIZATION OF WEIGHT VECTORS (AROW)

Since the passive-aggressive algorithm uses unbounded loss functions, it suffers a lot from outliers. This problem can be mitigated if a bounded loss function, such as the *ramp loss* (see Section 27.6.3) and the *Tukey loss* (see Section 25.4), is used. However, bounded loss functions are nonconvex and thus optimization becomes cumbersome in practice. In this section, a robust online learning algorithm called AROW [33] is introduced, which utilizes the sequential nature of online learning.

31.3.1 UNCERTAINTY OF PARAMETERS

In AROW, parameter $\boldsymbol{\theta}$ is not point-estimated, but its distribution is learned to take into account its uncertainty.

As a distribution of parameters, let us consider the Gaussian distribution whose probability density function is given as follows (Fig. 6.1):

$$(2\pi)^{-d/2}\det(\boldsymbol{\Sigma})^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right),$$

where $\det(\cdot)$ denotes the determinant, $\boldsymbol{\mu}$ denotes the expectation vector, and $\boldsymbol{\Sigma}$ denotes the variance-covariance matrix. Below, the Gaussian distribution with expectation vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$ is denoted by $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

AROW learns the expectation vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$ to minimize the following criterion:

$$J(\boldsymbol{\mu}) + \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}\boldsymbol{x} + \lambda \mathrm{KL}\Big(N(\boldsymbol{\mu}, \boldsymbol{\Sigma})\Big\|N(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})\Big). \tag{31.5}$$

The first term $J(\boldsymbol{\mu})$ denotes the loss for a new training sample $(\boldsymbol{x}, y)$ when parameter $\boldsymbol{\theta} = \boldsymbol{\mu}$ is used for prediction. The second term $\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}\boldsymbol{x}$ is the regularization term for variance-covariance matrix $\boldsymbol{\Sigma}$, which is adaptive to the training input vector $\boldsymbol{x}$. The third term $\lambda \mathrm{KL}(N(\boldsymbol{\mu}, \boldsymbol{\Sigma})\|N(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}}))$ controls the amount of change from the current solution, where $\lambda > 0$ denotes the *passiveness* parameter, $\widetilde{\boldsymbol{\mu}}$ and $\widetilde{\boldsymbol{\Sigma}}$ are the current solutions for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and $\mathrm{KL}(p\|q)$ denotes the *KL divergence* (see Section 14.2) from density p to density q:

$$\mathrm{KL}(p\|q) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}.$$

Passive-aggressive learning introduced in Section 31.2 used the Euclidean distance $\|\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}\|^2$ for controlling the amount of change from the current solution $\widetilde{\boldsymbol{\theta}}$. On the other hand, AROW uses the KL divergence for taking into account the uncertainty of parameters, which is expected to contribute to better controlling the amount of change from the current solution.

The KL divergence for Gaussian distributions can be expressed *analytically* as

$$\begin{aligned}&\mathrm{KL}\Big(N(\boldsymbol{\mu}, \boldsymbol{\Sigma})\Big\|N(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})\Big)\\ &= \frac{1}{2}\left(\log \frac{\det(\widetilde{\boldsymbol{\Sigma}})}{\det(\boldsymbol{\Sigma})} + \mathrm{tr}(\widetilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}) + (\boldsymbol{\mu} - \widetilde{\boldsymbol{\mu}})^\top \widetilde{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu} - \widetilde{\boldsymbol{\mu}}) - d\right),\end{aligned}$$

where d denotes the dimensionality of input vector $\boldsymbol{x}$.

Below, specific AROW algorithms for classification and regression are introduced.

31.3.2 CLASSIFICATION

Let us use the squared hinge loss (see Fig. 27.12) for expectation vector $\boldsymbol{\mu}$:

$$J(\boldsymbol{\mu}) = \frac{1}{2}\left(\max\left\{0, 1 - \boldsymbol{\mu}^\top \boldsymbol{x} y\right\}\right)^2.$$

Then setting the derivative of Eq. (31.5) with respect to $\boldsymbol{\mu}$ at zero yields that the solution $\widehat{\boldsymbol{\mu}}$ satisfies

$$\widehat{\boldsymbol{\mu}} = \widetilde{\boldsymbol{\mu}} + y \max(0, 1 - \widetilde{\boldsymbol{\mu}}^\top \boldsymbol{x} y)\widetilde{\boldsymbol{\Sigma}}\boldsymbol{x}/\beta,$$

where $\beta = \boldsymbol{x}^\top \widetilde{\boldsymbol{\Sigma}}\boldsymbol{x} + \lambda$.

```
1. Initialize $\boldsymbol{\mu} \longleftarrow \mathbf{0}$ and $\boldsymbol{\Sigma} \longleftarrow \boldsymbol{I}$.
2. Update parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as follows, if the margin $m = \boldsymbol{\mu}^\top \boldsymbol{x} y$ for a new
   training sample $(\boldsymbol{x}, y)$ satisfies $m < 1$:
       $\boldsymbol{\mu} \longleftarrow \boldsymbol{\mu} + y \max(0, 1 - m)\boldsymbol{\Sigma}\boldsymbol{x}/\beta$ and $\boldsymbol{\Sigma} \longleftarrow \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{\Sigma}/\beta$,
   where $\beta = \boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x} + \lambda$.
3. Go to Step 2.
```

FIGURE 31.6

Algorithm of AROW classification.

The matrix derivative formulas (see Fig. 12.3),

$$\frac{\partial}{\partial \boldsymbol{\Sigma}} \log(\det(\boldsymbol{\Sigma})) = \boldsymbol{\Sigma}^{-1} \quad \text{and} \quad \frac{\partial}{\partial \boldsymbol{\Sigma}} \text{tr}\left(\widetilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\right) = \widetilde{\boldsymbol{\Sigma}}^{-1},$$

allow us to compute the partial derivative of Eq. (31.5) with respect to $\boldsymbol{\Sigma}$, and setting it at zero gives the solution $\widehat{\boldsymbol{\Sigma}}$ analytically as

$$\widehat{\boldsymbol{\Sigma}}^{-1} = \widetilde{\boldsymbol{\Sigma}}^{-1} - \boldsymbol{x}\boldsymbol{x}^\top/\lambda.$$

Further applying the *matrix inversion lemma* (see Fig. 23.12) yields

$$\widehat{\boldsymbol{\Sigma}} = \widetilde{\boldsymbol{\Sigma}} - \widetilde{\boldsymbol{\Sigma}}\boldsymbol{x}\boldsymbol{x}^\top\widetilde{\boldsymbol{\Sigma}}/\beta.$$

This expression allows us to directly obtain the solution $\widehat{\boldsymbol{\Sigma}}$ without computing its inverse explicitly.

The algorithm of AROW classification is summarized in Fig. 31.6. When the dimensionality d of the input vector $\boldsymbol{x}$ is large, updating the $d \times d$ variance-covariance matrix $\boldsymbol{\Sigma}$ is computationally expensive. A practical approach to mitigating this problem is to only maintain the diagonal elements of $\boldsymbol{\Sigma}$ and regard all off-diagonal elements as zero, which corresponds to only considering the Gaussian distribution whose principal axes of the elliptic contour lines agree with the coordinate axes (see Fig. 6.1).

A MATLAB code for AROW classification is provided in Fig. 31.7, and its behavior is illustrated in Fig. 31.8. This shows that AROW suppresses the influence of outliers better than passive-aggressive learning.

31.3.3 REGRESSION

Let us use the squared loss for the expectation vector $\boldsymbol{\mu}$:

$$J(\boldsymbol{\mu}) = \frac{1}{2}\left(y - \boldsymbol{\mu}^\top\boldsymbol{x}\right)^2.$$

```
n=50; x=[randn(1,n/2)-15 randn(1,n/2)-5; randn(1,n)]';
y=[ones(n/2,1); -ones(n/2,1)]; x(1:2,1)=x(1:2,1)+10;
x(:,3)=1; p=randperm(n); x=x(p,:); y=y(p);
mu=zeros(3,1); S=eye(3); l=1;
for i=1:n
  xi=x(i,:)'; yi=y(i); z=S*xi; b=xi'*z+l; m=yi*mu'*xi;
  if m<1, mu=mu+yi*(1-m)*z/b; S=S-z*z'/b; end
end
figure(1); clf; hold on; axis([-20 0 -2 2]);
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==-1,1),x(y==-1,2),'rx');
plot([-20 0],-(mu(3)+[-20 0]*mu(1))/mu(2),'k-');
```

FIGURE 31.7

MATLAB code for AROW classification.

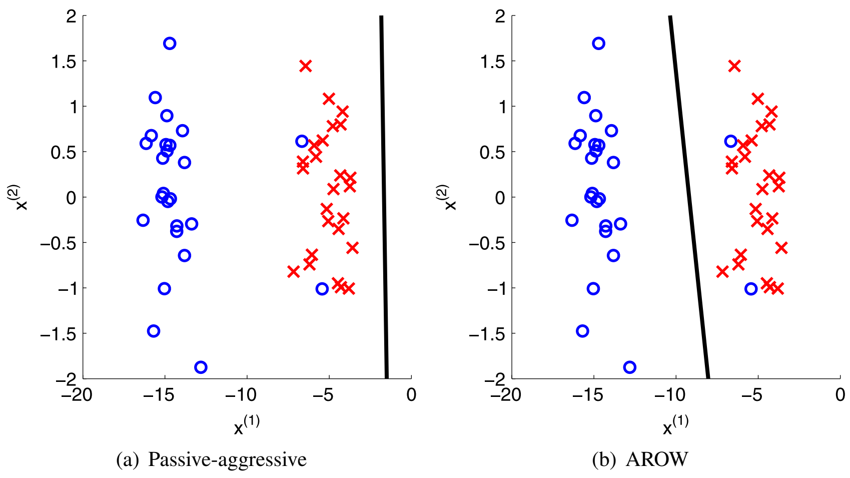

FIGURE 31.8

Examples of passive-aggressive and AROW classifications.

Then, with a similar derivation to the classification case, the update formula for AROW regression can be obtained analytically as

$$\boldsymbol{\mu} \longleftarrow \boldsymbol{\mu} + (y - \boldsymbol{\mu}^\top \boldsymbol{x})\boldsymbol{\Sigma}\boldsymbol{x}/\beta \quad \text{and} \quad \boldsymbol{\Sigma} \longleftarrow \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{\Sigma}/\beta,$$

where $\beta = \boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x} + \lambda$. This method is also referred to as *recursive LS*. When $\boldsymbol{\Sigma}$ is fixed to the identity matrix and setting $\boldsymbol{\theta} = \boldsymbol{\mu}$, AROW regression is reduced to

```
n=50; N=1000; x=linspace(-3,3,n)'; X=linspace(-3,3,N)';
pix=pi*x; y=sin(pix)./(pix)+0.1*x+0.05*randn(n,1);
hh=2*0.3^2; m=randn(n,1); S=eye(n); l=1;
for j=1:100
  for i=1:n
    ki=exp(-(x-x(i)).^2/hh); Sk=S*ki;
    b=ki'*Sk+l; m=m+Sk*(y(i)-ki'*m)/b; S=S-Sk*Sk'/b;
end, end
K=exp(-(repmat(X.^2,1,n)+repmat(x.^2',N,1)-2*X*x')/hh);
F=K*m;
figure(1); clf; hold on; axis([-2.8 2.8 -0.5 1.2]);
plot(X,F,'g-'); plot(x,y,'bo');
```

FIGURE 31.9

MATLAB code for AROW regression.

passive-aggressive regression. Thus, AROW can be regarded as a natural extension of passive-aggressive learning.

A MATLAB code for AROW regression is provided in Fig. 31.9.