CHAPTER

33 SEMISUPERVISED LEARNING

CHAPTER CONTENTS

Supervised learning is performed based on input-output paired training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$. However, gathering many input-output paired samples is often expensive in practical applications. On the other hand, *input-only* samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ can be easily collected abundantly. For example, in web page classification, class label y_i (such as "sport", "computer", and "politics") should be *manually* given after carefully investigating web page $\boldsymbol{x}_i$, which requires a huge amount of human labor. On the other hand, input-only web pages $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ can be automatically collected by crawlers.

In this chapter, methods of *semisupervised learning* [28, 101] are introduced, which utilize input-only samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ in addition to input-output paired samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.

33.1 MANIFOLD REGULARIZATION

In this section, a method of semisupervised learning based on *manifold regularization* is introduced.

33.1.1 MANIFOLD STRUCTURE BROUGHT BY INPUT SAMPLES

Supervised learning from input-output paired samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ can be regarded as estimating the conditional density $p(y|\boldsymbol{x})$. On the other hand, unsupervised

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00044-3

Copyright © 2016 by Elsevier Inc. All rights of reproduction in any form reserved.

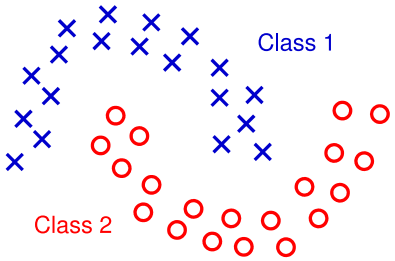


FIGURE 33.1

Semisupervised classification. Samples in the same cluster are assumed to belong to the same class.

learning from input-only samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ can be regarded as estimating the marginal density $p(\boldsymbol{x})$. Thus, without any assumption, input-only samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ do not help improve the estimation of the conditional density $p(y|\boldsymbol{x})$. For this reason, semisupervised learning imposes a certain assumption between $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ and utilizes the estimation of $p(\boldsymbol{x})$ to improve the accuracy of estimating $p(y|\boldsymbol{x})$.

Below, a semisupervised learning method based on a *manifold* assumption is introduced. Mathematically, a manifold is a topological space that can be locally approximated by Euclidean space. On the other hand, a manifold is just regarded as a local region in the context of semisupervised learning. More specifically, the manifold assumption means that input samples appear only on manifolds and output values change *smoothly* on the manifolds. In the case of classification, this means that samples in the same cluster belong to the same class (Fig. 33.1).

The Gaussian kernel model introduced in Section 21.2 actually utilizes this manifold assumption (see Fig. 21.5):

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

That is, by locating smooth Gaussian functions on input samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$, a smooth function over the input manifold can be learned. In semisupervised learning, the above model may be augmented to locate Gaussian kernels also on input-only samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n+n'} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j). \tag{33.1}$$

The parameters in model (33.1) are learned so that output at input samples, $\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}_{i=1}^{n+n'}$, is similar to each other. For example, in the case of ℓ_2-regularized

LS, the optimization problem is given as

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 + \frac{\nu}{4} \sum_{i,i'=1}^{n+n'} W_{i,i'} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}) \right)^2 \right], \tag{33.2}$$

where the first and second terms correspond to ℓ_2-regularized LS and the third term is called the *Laplacian regularizer*, following the terminology in *spectral graph theory* [29]. $\nu \geq 0$ is the regularization parameter for semisupervised learning that controls the smoothness on the manifolds. $W_{i,i'} \geq 0$ denotes the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$, which takes a large value if $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$ are similar, and a small value if $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$ are dissimilar. Popular choices of the similarity measure are described in Fig. 35.8. Below the similarity matrix $\boldsymbol{W}$ is assumed to be symmetric (i.e. $W_{i,i'} = W_{i',i}$).

33.1.2 COMPUTING THE SOLUTION

Here, how to compute the solution of Laplacian-regularized LS is explained. Let $\boldsymbol{D}$ be the diagonal matrix whose diagonal elements are given by the row-sums of matrix $\boldsymbol{W}$:

$$\boldsymbol{D} = \mathrm{diag}\left(\sum_{i=1}^{n+n'} W_{1,i}, \ldots, \sum_{i=1}^{n+n'} W_{n+n',i} \right),$$

and let $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$. Then the third term in Eq. (33.2) can be rewritten as

$$\begin{aligned} &\sum_{i,i'=1}^{n+n'} W_{i,i'} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}) \right)^2 \\ &= \sum_{i=1}^{n+n'} D_{i,i} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^2 - 2 \sum_{i,i'=1}^{n+n'} W_{i,i'} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}) + \sum_{i'=1}^{n+n'} D_{i',i'} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^2 \\ &= 2 \sum_{i,i'=1}^{n+n'} L_{i,i'} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}). \end{aligned}$$

Thus, Eq. (33.2) for kernel model (33.1) can be reduced to the generalized ℓ_2-regularized LS as follows:

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{i=1}^{n} \left(\sum_{j=1}^{n+n'} \theta_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - y_i \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n+n'} \theta_j^2 + \frac{\nu}{2} \sum_{j,j'=1}^{n+n'} \theta_j \theta_{j'} \sum_{i,i'=1}^{n+n'} L_{i,i'} K(\boldsymbol{x}_i, \boldsymbol{x}_j) K(\boldsymbol{x}_{i'}, \boldsymbol{x}_{j'}) \right].$$

This can be compactly rewritten as

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \|\boldsymbol{K}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 + \frac{\nu}{2} \boldsymbol{\theta}^\top \boldsymbol{K}\boldsymbol{L}\boldsymbol{K}\boldsymbol{\theta} \right],$$

where

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_{n+n'}) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{n+n'}, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{n+n'}, \boldsymbol{x}_{n+n'}) \end{pmatrix},$$
$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n, \theta_{n+1}, \ldots, \theta_{n+n'})^\top,$$
$$\boldsymbol{y} = (y_1, \ldots, y_n, \underbrace{0, \ldots, 0}_{n'})^\top.$$

Note that the vector $\boldsymbol{y}$ defined above contains n' zeros since $\{y_i\}_{i=n+1}^{n+n'}$ are not available. The solution $\widehat{\boldsymbol{\theta}}$ can be obtained analytically as

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{K}^2 + \lambda \boldsymbol{I} + \nu \boldsymbol{K}\boldsymbol{L}\boldsymbol{K})^{-1} \boldsymbol{K}\boldsymbol{y}.$$

A MATLAB code for Laplacian-regularized LS is provided in Fig. 33.2, and its behavior is illustrated in Fig. 33.3. This is an extremely difficult classification problem where only two labeled training samples are available. Laplacian-regularized LS gives a decision boundary that separates two point clouds, which is appropriate when the cluster assumption holds. On the other hand, ordinary LS gives a decision boundary that goes through the middle of the two labeled training samples, which is not reasonable under the cluster assumption.

In the above explanation, Laplacian regularization was applied to LS. However, the idea of Laplacian regularization is generic and it can be combined with various regression and classification techniques introduced in Part 4.

33.2 COVARIATE SHIFT ADAPTATION

The Laplace regularization method introduced above utilized the manifold structure of input samples. In this section, a semisupervised learning method called *covariate shift* adaptation is introduced [101], which explicitly takes into account the probability distributions of $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$. A covariate is another name for an input variable, and covariate shift refers to the situation where $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ follow different probability distributions, but the conditional density $p(y|\boldsymbol{x})$ is unchanged.

33.2.1 IMPORTANCE WEIGHTED LEARNING

An example of covariate shift in regression is illustrated in Fig. 33.4, where the learning target function $f(x)$ is unchanged. However, $\{x_i\}_{i=1}^n$ are distributed

```
n=200; a=linspace(0,pi,n/2);
u=-10*[cos(a)+0.5 cos(a)-0.5]'+randn(n,1);
v=10*[sin(a) -sin(a)]'+randn(n,1);
x=[u v]; y=zeros(n,1); y(1)=1; y(n)=-1;
x2=sum(x.^2,2); hh=2*1^2;
k=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x')/hh); w=k;
t=(k^2+1*eye(n)+10*k*(diag(sum(w))-w)*k)\(k*y);

m=100; X=linspace(-20,20,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; axis([-20 20 -20 20]);
colormap([1 0.7 1; 0.7 1 1]);
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');
plot(x(y==0,1),x(y==0,2),'k.');
```

FIGURE 33.2

MATLAB code for Laplacian-regularized LS.

around $x = 1$, while $\{x'_{i'}\}_{i'=1}^{n'}$ are distributed around $x = 2$. Thus, this is a (weak) *extrapolation* problem.

Fig. 33.6(a) illustrates the function obtained by fitting the straight line model,

$$f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x, \tag{33.3}$$

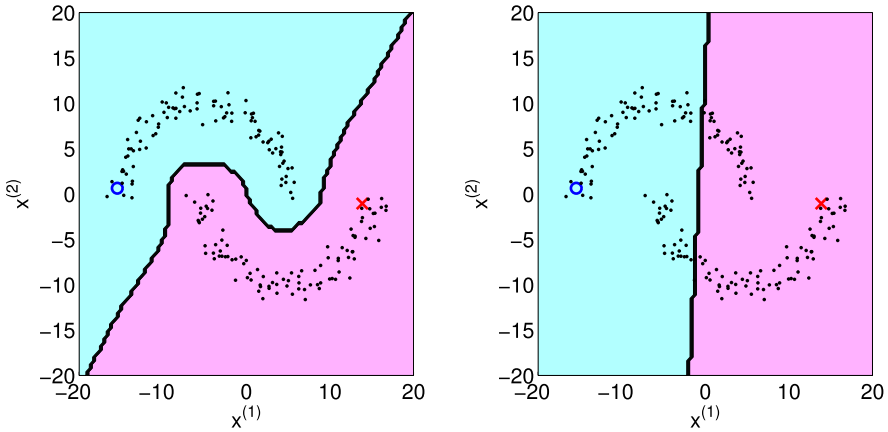
to samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ by ordinary LS:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2.$$

This shows that, although the samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ are appropriately fitted by LS, prediction of output values at $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ is poor.

In such a covariate shift situation, it intuitively seems that only using samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ *near* $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ can provide good prediction of output values at $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$. This intuitive idea can be more formally realized by *importance weighting*. More specifically, *importance weighted LS* is given by

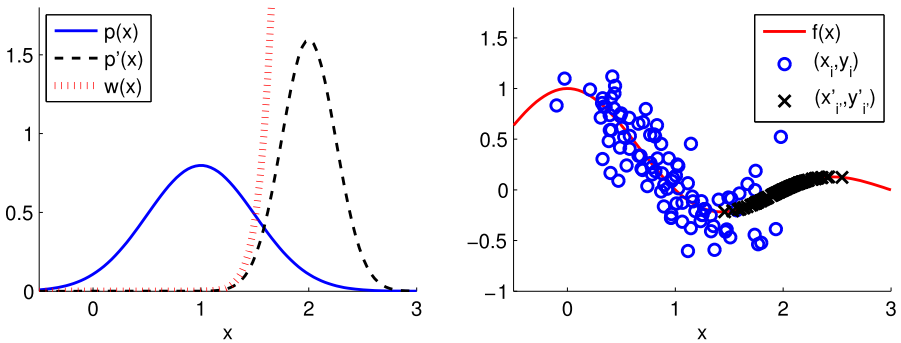
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} w(\boldsymbol{x}_i) \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2,$$



(a) Laplacian-regularized LS (b) Ordinary LS

FIGURE 33.3

Examples of Laplacian-regularized LS compared with ordinary LS. Dots denote unlabeled training samples.



(a) Input densities and importance (b) Learning target function and training samples

FIGURE 33.4

Covariate shift in regression. Input distributions change, but the input-output relation is unchanged.

where $w(\boldsymbol{x})$ is called the importance function which is defined as the ratio of probability density $p'(\boldsymbol{x})$ for $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ and probability density $p(\boldsymbol{x})$ for $\{\boldsymbol{x}_i\}_{i=1}^{n}$:

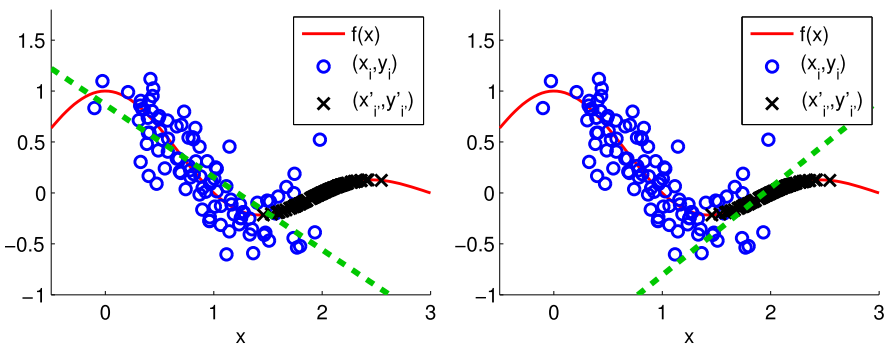
$$w(\boldsymbol{x}) = \frac{p'(\boldsymbol{x})}{p(\boldsymbol{x})}.$$

```
n=100; u=randn(n,1)/4+2; x=randn(n,1)/2+1;
w=2*exp(-8*(x-2).^2+2*(x-1).^2); %w=ones(n,1);

y=sin(pi*x)./(pi*x)+0.1*randn(n,1);
x(:,2)=1; t=(x'*(repmat(w,1,2).*x))\(x'*(w.*y));
X=linspace(-1,3,100); Y=sin(pi*X)./(pi*X);
u(:,2)=1; v=u*t;
figure(1); clf; hold on;
plot(x(:,1),y,'bo'); plot(X,Y,'r-'); plot(u(:,1),v,'kx');
```

FIGURE 33.5

MATLAB code for importance weighted LS.



(a) Ordinary LS (b) Importance weighted LS

FIGURE 33.6

Example of LS learning under covariate shift. The dashed lines denote learned functions.

This importance weighted learning is based on the idea of *importance sampling* (see Section 19.2), which approximates the expectation with respect to $p'(\boldsymbol{x})$ by importance weighted average with respect to $p(\boldsymbol{x})$:

$$\int g(\boldsymbol{x})p'(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int g(\boldsymbol{x})\frac{p'(\boldsymbol{x})}{p(\boldsymbol{x})}p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{x}_i)w(\boldsymbol{x}_i).$$

A MATLAB code of importance weighted LS for straight line model (33.3) is provided in Fig. 33.5, and its behavior is illustrated in Fig. 33.6(b). This shows that importance weighting contributes to improving the accuracy of predicting output values at $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$.

In the above example, importance weighting was applied to LS. However, the idea of importance weighting is generic and it can be combined with various regression and classification techniques introduced in Part 4.

33.2.2 RELATIVE IMPORTANCE WEIGHTED LEARNING

As illustrated in Fig. 33.6(b), prediction performance under covariate shift can be improved by importance weighted learning.

Fig. 33.4(a) illustrates the importance function $w(x)$, which is monotone increasing as x grows. This means that, among all training samples $\{(x_i, y_i)\}_{i=1}^n$, only a few of them with large x have large importance values and the importance values of others are negligibly small. Thus, importance weighted learning is actually performed only from a few training samples, which can be unstable in practice.

This instability is caused by the fact that the importance function $w(\boldsymbol{x})$ takes large values for some $\boldsymbol{x}$. Thus, the instability problem is expected to be mitigated if a smooth variant of the importance function, such as the *relative importance* [121] defined as

$$w_\beta(\boldsymbol{x}) = \frac{p'(\boldsymbol{x})}{\beta p'(\boldsymbol{x}) + (1-\beta)p(\boldsymbol{x})},$$

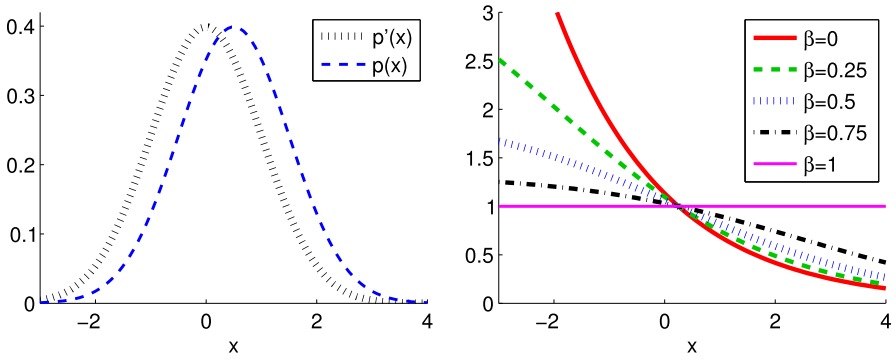
is employed, where $\beta \in [0,1]$ controls the smoothness. The relative importance function $w_\beta(\boldsymbol{x})$ is reduced to the ordinary importance function $p'(\boldsymbol{x})/p(\boldsymbol{x})$ if $\beta = 0$. It gets smoother if β is increased, and it yields the uniform weight $w_\beta(\boldsymbol{x}) = 1$ if $\beta = 1$ (Fig. 33.7). Since the importance is always non-negative, the relative importance is no larger than $1/\beta$:

$$w_\beta(\boldsymbol{x}) = \frac{1}{\beta + (1-\beta)\frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}} \leq \frac{1}{\beta}.$$

33.2.3 IMPORTANCE WEIGHTED CROSS VALIDATION

The performance of relative importance weighted learning depends on the choice of smoothness β. Also, choice of model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and regularization parameters significantly affects the final performance.

In Section 23.3, a model selection method based on *cross validation* was introduced, which estimates the prediction error for test samples. Actually, the validity of cross validation is ensured only when $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ follow the same probability distributions. Thus, performing cross validation under covariate shift can produce undesired solutions. Under covariate shift, a variant called *importance weighted cross validation* [102] is useful. The algorithm of importance weighted cross validation is summarized in Fig. 33.8, which is essentially the same as ordinary cross validation, but the validation error is computed with importance weights.



(a) Probability densities (b) Relative importance

FIGURE 33.7

Relative importance when $p'(x)$ is the Gaussian density with expectation 0 and variance 1 and $p(x)$ is the Gaussian density with expectation 0.5 and variance 1.

33.2.4 IMPORTANCE ESTIMATION

In (relative) importance weighted learning and cross validation, importance weights are needed. However, the importance function is usually unknown and only samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ are available. A naive way to estimate the importance function $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$ is to separately estimate density $p'(\boldsymbol{x})$ from $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ and $p(\boldsymbol{x})$ from $\{\boldsymbol{x}_i\}_{i=1}^{n}$ and then compute the ratio of estimated densities. However, such a two-step approach is unreliable because division by an estimated density can magnify the estimation error significantly. Here, a direct importance estimator without going through density estimation is introduced.

Let us model the relative importance function $w_\beta(\boldsymbol{x})$ by a linear-in-parameter model:

$$w_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{j=1}^{b} \alpha_j \psi_j(\boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\psi}(\boldsymbol{x}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_b)^\top$ is the parameter vector and $\boldsymbol{\psi}(\boldsymbol{x}) = (\psi_1(\boldsymbol{x}), \ldots, \psi_b(\boldsymbol{x}))^\top$ is the vector of basis functions. The parameter $\boldsymbol{\alpha}$ is learned so that the following $J(\boldsymbol{\alpha})$ is minimized:

$$\begin{aligned} J(\boldsymbol{\alpha}) &= \frac{1}{2} \int \Big(w_{\boldsymbol{\alpha}}(\boldsymbol{x}) - w_\beta(\boldsymbol{x})\Big)^2 \Big(\beta p'(\boldsymbol{x}) + (1-\beta) p(\boldsymbol{x})\Big) \mathrm{d}\boldsymbol{x} \\ &= \frac{1}{2} \int \boldsymbol{\alpha}^\top \boldsymbol{\psi}(\boldsymbol{x}) \boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{\alpha} \Big(\beta p'(\boldsymbol{x}) + (1-\beta) p(\boldsymbol{x})\Big) \mathrm{d}\boldsymbol{x} \\ &\quad - \int \boldsymbol{\alpha}^\top \boldsymbol{\psi}(\boldsymbol{x}) p'(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + C, \end{aligned}$$

1. Prepare candidates of models: $\{\mathcal{M}_j\}_j$.
2. Split training samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into t disjoint subsets of (approximately) the same size: $\{\mathcal{D}_\ell\}_{\ell=1}^t$.
3. For each model candidate $\mathcal{M}_j$
   (a) For each split $\ell = 1, \ldots, t$
      i. Obtain learned function $\widehat{f}_j^{(\ell)}(\boldsymbol{x})$ using model $\mathcal{M}_j$ from all training samples without $\mathcal{D}_\ell$.
      ii. Compute the average prediction error $\widehat{G}_j^{(\ell)}$ of $\widehat{f}_j^{(\ell)}(\boldsymbol{x})$ for holdout samples $\mathcal{D}_\ell$.
      - Regression (squared loss):

$$\widehat{G}_j^{(\ell)} = \frac{1}{|\mathcal{D}_\ell|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_\ell} w(\boldsymbol{x})\left(y - \widehat{f}_j^{(\ell)}(\boldsymbol{x})\right)^2,$$

        where $|\mathcal{D}_\ell|$ denotes the number of elements in the set $\mathcal{D}_\ell$.
      - Classification (0/1-loss):

$$\widehat{G}_j^{(\ell)} = \frac{1}{|\mathcal{D}_\ell|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_\ell} \frac{w(\boldsymbol{x})}{2}\left(1 - \operatorname{sign}\left(\widehat{f}_j^{(\ell)}(\boldsymbol{x})y\right)\right).$$

   (b) Compute the average prediction error $\widehat{G}_j$ over all t splits:

$$\widehat{G}_j = \frac{1}{t}\sum_{\ell=1}^t \widehat{G}_j^{(\ell)}.$$

4. Choose the model $\mathcal{M}_{\widehat{j}}$ that minimizes the average prediction error:

$$\widehat{j} = \operatorname*{argmin}_j \widehat{G}_j.$$

5. Obtain the final function approximator using chosen model $\mathcal{M}_{\widehat{j}}$ from all training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.

FIGURE 33.8

Algorithm of importance weighted cross validation.

where $C = \frac{1}{2}\int w_\beta(\boldsymbol{x})p'(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ is independent of $\boldsymbol{\alpha}$ and thus can be ignored. Approximating the expectations by sample averages and adding the ℓ_2-regularizer yield the following optimization problem:

$$\min_{\boldsymbol{\alpha}} \left[\frac{1}{2} \boldsymbol{\alpha}^\top \widehat{\boldsymbol{G}}_\beta \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \widehat{\boldsymbol{h}} + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|^2 \right],$$

where

$$\widehat{\boldsymbol{G}}_\beta = \frac{\beta}{n'} \sum_{i'=1}^{n'} \boldsymbol{\psi}(\boldsymbol{x}'_{i'}) \boldsymbol{\psi}(\boldsymbol{x}'_{i'})^\top + \frac{1-\beta}{n} \sum_{i=1}^{n} \boldsymbol{\psi}(\boldsymbol{x}_i) \boldsymbol{\psi}(\boldsymbol{x}_i)^\top,$$

$$\widehat{\boldsymbol{h}} = \frac{1}{n'} \sum_{i'=1}^{n'} \boldsymbol{\psi}(\boldsymbol{x}'_{i'}).$$

The minimizer $\widehat{\boldsymbol{\alpha}}$ can be obtained analytically as

$$\widehat{\boldsymbol{\alpha}} = \left(\widehat{\boldsymbol{G}} + \lambda \boldsymbol{I} \right)^{-1} \widehat{\boldsymbol{h}}.$$

This method is called *LS relative density ratio estimation* [121]. The regularization parameter λ and parameters included in basis functions $\boldsymbol{\psi}$ can be optimized by cross validation with respect to the squared error J.

A MATLAB code of LS relative density ratio estimation for the Gaussian kernel model,

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

is provided in Fig. 33.9, and its behavior is illustrated in Fig. 33.10. This shows that the true relative importance function is nicely estimated.

33.3 CLASS-BALANCE CHANGE ADAPTATION

In the previous section, semisupervised learning methods for covariate shift adaptation were introduced, which can be naturally applied in regression. On the other hand, in classification, *class-balance change* is a natural situation, where the class-prior probabilities differ in $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ (see Fig. 33.11), but the class-conditional probability $p(\boldsymbol{x}|y)$ remains unchanged. In this section, adaptation methods for class-balance change are introduced.

33.3.1 CLASS-BALANCE WEIGHTED LEARNING

The bias caused by class-balance change can be canceled by *class-balance weighted learning*.

More specifically, the class-prior ratio $p'(y)/p(y)$ is used as a weighting factor, where $p(y)$ and $p'(y)$ are the class-prior probabilities for $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$, respectively. For example, in the case of LS, the learning criterion is given by

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

```
n=300; x=randn(n,1); y=randn(n,1)+0.5;
hhs=2*[1 5 10].^2; ls=10.^[-3 -2 -1]; m=5; b=0.5;
x2=x.^2; xx=repmat(x2,1,n)+repmat(x2',n,1)-2*x*x';
y2=y.^2; yx=repmat(y2,1,n)+repmat(x2',n,1)-2*y*x';
u=mod(randperm(n),m)+1; v=mod(randperm(n),m)+1;

for hk=1:length(hhs)
  hh=hhs(hk); k=exp(-xx/hh); r=exp(-yx/hh);
  for i=1:m
    ki=k(u~=i,:); ri=r(v~=i,:); h=mean(ki)';
    kc=k(u==i,:); rj=r(v==i,:);
    G=b*ki'*ki/sum(u~=i)+(1-b)*ri'*ri/sum(v~=i);
    for lk=1:length(ls)
      l=ls(lk); a=(G+l*eye(n))\h; kca=kc*a;
      g(hk,lk,i)=b*mean(kca.^2)+(1-b)*mean((rj*a).^2);
      g(hk,lk,i)=g(hk,lk,i)/2-mean(kca);
end, end, end
[gl,ggl]=min(mean(g,3),[],2); [ghl,gghl]=min(gl);
L=ls(ggl(gghl)); HH=hhs(gghl);
k=exp(-xx/HH); r=exp(-yx/HH);
s=r*((b*k'*k/n+(1-b)*r'*r/n+L*eye(n))\(mean(k)'));
figure(1); clf; hold on; plot(y,s,'rx');
```

FIGURE 33.9

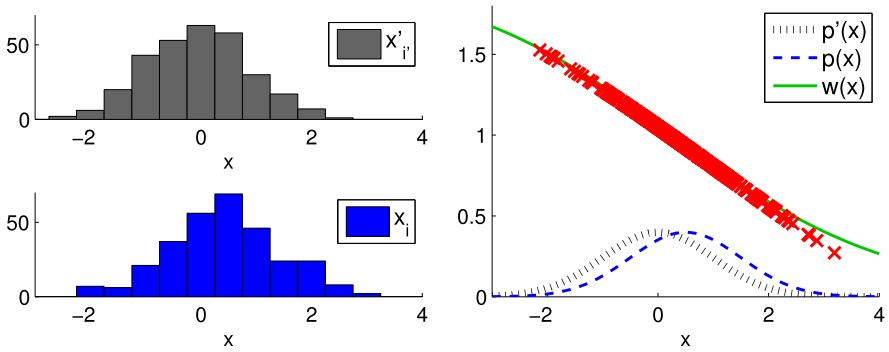
MATLAB code for LS relative density ratio estimation for Gaussian kernel model.

which is called *class-balance weighted LS*. Beyond LS, this class-balance change adaptation technique can be applied to various classification methods introduced in Part 4.

Model selection of class-balance weighted learning may be carried out by *class-balance weighted cross validation*, which is essentially the same as *importance weighted cross validation* introduced in Section 33.2.3, but $p'(y_i)/p(y_i)$ is used as a weight.

33.3.2 CLASS-BALANCE ESTIMATION

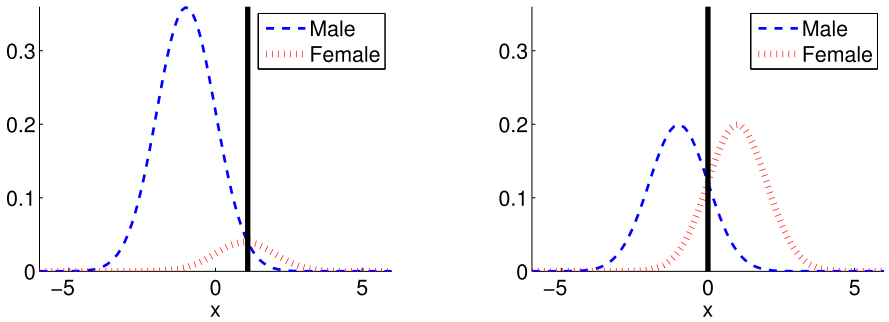
To use class-balance weighted learning, the class-prior probabilities $p(y)$ and $p'(y)$ are needed, which are often unknown in practice. For labeled samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, estimating the class prior $p(y)$ is straightforward by n_y/n, where n_y denotes the number of samples in class y. However, $p'(y)$ cannot be estimated naively because of



(a) Input samples (b) Relative importance ($\beta = 0.5$)

FIGURE 33.10

Example of LS relative density ratio estimation. ×'s in the right plot show estimated relative importance values at $\{\boldsymbol{x}_i\}_{i=1}^{n}$.



(a) Labeled samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ (b) Unlabeled samples $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$

FIGURE 33.11

Class-balance change, which affects the decision boundary.

lack of output values $\{y'_{i'}\}_{i'=1}^{n'}$ for $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$. Below, a practical estimator of $p'(y)$ is introduced [38].

The basic idea of estimating $p'(y)$ is to fit a mixture $q_{\boldsymbol{\pi}}(\boldsymbol{x})$ of classwise input densities $p(\boldsymbol{x}|y)$ to $p'(\boldsymbol{x})$ (Fig. 33.12):

$$q_{\boldsymbol{\pi}}(\boldsymbol{x}) = \sum_{y=1}^{c} \pi_y p(\boldsymbol{x}|y),$$

where c denotes the number of classes and the coefficient π_y corresponds to $p'(y)$.

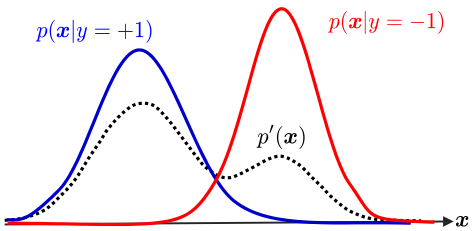


FIGURE 33.12

Class-prior estimation by distribution matching.

Matching of $q_{\boldsymbol{\pi}}$ to p' can be performed, for example, under the *KL divergence* (see Section 14.2),

$$\mathrm{KL}(p' \| q_{\boldsymbol{\pi}}) = \int p'(\boldsymbol{x}) \log \frac{p'(\boldsymbol{x})}{q_{\boldsymbol{\pi}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x},$$

or the *L_2-distance*:

$$L_2(p', q_{\boldsymbol{\pi}}) = \int \left(p'(\boldsymbol{x}) - q_{\boldsymbol{\pi}}(\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{x}.$$

The KL divergence can be accurately estimated by *KL density ratio estimation* explained in Section 38.3, while the L_2-distance can be accurately estimated by *LS density difference estimation* introduced in Section 39.1.3.

Another useful distance measure is the *energy distance* [106], which is the weighted squared distance between characteristic functions:

$$D_{\mathrm{E}}(p', q_{\boldsymbol{\pi}}) = \int_{\mathbb{R}^d} \|\boldsymbol{\varphi}_{p'}(\boldsymbol{t}) - \boldsymbol{\varphi}_{q_{\boldsymbol{\pi}}}(\boldsymbol{t})\|^2 \left(\frac{\pi^{\frac{d+1}{2}}}{\Gamma\left(\frac{d+1}{2}\right)} \|\boldsymbol{t}\|^{d+1} \right)^{-1} \mathrm{d}\boldsymbol{t},$$

where $\|\cdot\|$ denotes the Euclidean norm, $\boldsymbol{\varphi}_p$ denotes the *characteristic function* (see Section 2.4.3) of p, $\Gamma(\cdot)$ is the *gamma function* (see Section 4.3), and d denotes the dimensionality of $\boldsymbol{x}$. Thanks to the careful design of the weight function, the energy distance can be equivalently expressed as

$$\begin{aligned} D_{\mathrm{E}}(p', q_{\boldsymbol{\pi}}) &= 2\mathbb{E}_{\boldsymbol{x}' \sim p', \boldsymbol{x} \sim q_{\boldsymbol{\pi}}} \|\boldsymbol{x}' - \boldsymbol{x}\| - \mathbb{E}_{\boldsymbol{x}', \widetilde{\boldsymbol{x}}' \sim p'} \|\boldsymbol{x}' - \widetilde{\boldsymbol{x}}'\| - \mathbb{E}_{\boldsymbol{x}, \widetilde{\boldsymbol{x}} \sim q_{\boldsymbol{\pi}}} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\| \\ &= 2\boldsymbol{\pi}^\top \boldsymbol{b} - \boldsymbol{\pi}^\top \boldsymbol{A} \boldsymbol{\pi} + C, \end{aligned}$$

where $\mathbb{E}_{\boldsymbol{x}' \sim p'}$ denotes the expectation with respect to $\boldsymbol{x}'$ following density p' and C is a constant independent of $\boldsymbol{\pi}$. $\boldsymbol{A}$ is the $c \times c$ symmetric matrix and $\boldsymbol{b}$ is the c-dimensional vector defined as

$$A_{y, \widetilde{y}} = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|y), \widetilde{\boldsymbol{x}} \sim p(\boldsymbol{x}|\widetilde{y})} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\|,$$

```
x=[[randn(90,1)-2; randn(10,1)+2] 2*randn(100,1)];
x(:,3)=1; y=[ones(90,1); 2*ones(10,1)]; n=length(y);
X=[[randn(10,1)-2; randn(90,1)+2] 2*randn(100,1)];
X(:,3)=1; Y=[ones(10,1); 2*ones(90,1)]; N=length(Y);
x2=sum(x.^2,2); X2=sum(X.^2,2);
xx=sqrt(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x');
xX=sqrt(repmat(x2,1,N)+repmat(X2',n,1)-2*x*X');
for i=1:2
  s(i)=sum(y==i)/n; b(i)=mean(mean(xX(y==i,:)));
  for j=1:2
    A(i,j)=mean(mean(xx(y==i,y==j)));
end, end
v=(A(1,2)-A(2,2)-b(1)+b(2))/(2*A(1,2)-A(1,1)-A(2,2));
v=min(1,max(0,v)); v(2)=1-v; w=v(y)./s(y); z=2*y-3;
u=x\z; t=(x'*(repmat(w',1,size(x,2)).*x))\(x'*(w'.*z));
figure(1); clf; hold on
plot([-5 5],-(u(3)+[-5 5]*u(1))/u(2),'k--');
plot([-5 5],-(t(3)+[-5 5]*t(1))/t(2),'g-');
plot(X(Y==1,1),X(Y==1,2),'bo');
plot(X(Y==2,1),X(Y==2,2),'rx');
legend('Unweighted','Weighted'); axis([-5 5 -10 10])
```

FIGURE 33.13

MATLAB code for class-balance weighted LS.

$$b_y = \mathbb{E}_{\boldsymbol{x}' \sim p', \boldsymbol{x} \sim p(\boldsymbol{x}|y)} \|\boldsymbol{x}' - \boldsymbol{x}\|.$$

Although $D_{\mathrm{E}}(p', q_{\boldsymbol{\pi}})$ is a concave function with respect to $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_c)^\top$, it is a convex function with respect to $\pi_1, \ldots, \pi_{c-1}$ for $\pi_c = 1 - \sum_{y=1}^{c-1} \pi_y$ and thus its minimizer can be easily obtained [61]. For example, when $c = 2$ with $\pi_1 = \pi$ and $\pi_2 = 1 - \pi$, $D_{\mathrm{E}}(p', q_{\boldsymbol{\pi}})$ can be expressed as a function of π up to a constant as

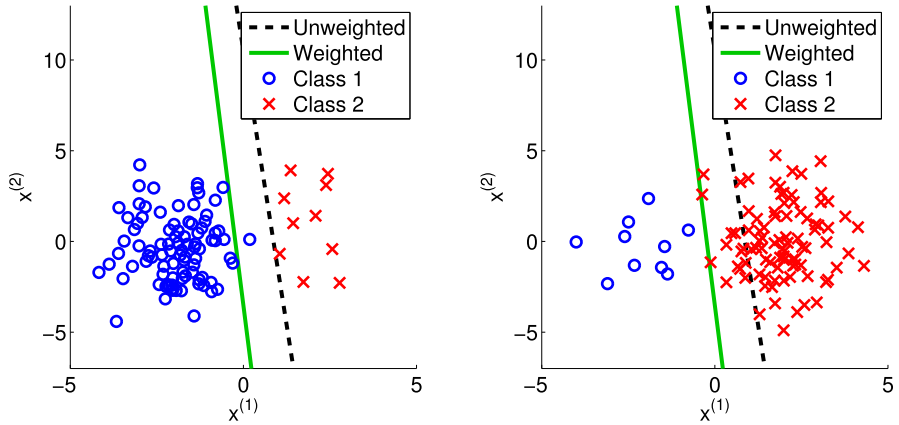
$$J(\pi) = a\pi^2 - 2b\pi,$$

where

$$a = 2A_{1,2} - A_{1,1} - A_{2,2} = D_{\mathrm{E}}(p(\boldsymbol{x}|y=1), p(\boldsymbol{x}|y=2)) \geq 0,$$
$$b = A_{1,2} - A_{2,2} - b_1 + b_2.$$

Since $J(\pi)$ is convex with respect to π, its minimizer is given analytically as $\min(1, \max(0, b/a))$. Note that $A_{y,\widetilde{y}}$ and b_y can be empirically approximated as

$$\widehat{A}_{y,\widetilde{y}} = \frac{1}{n_y n_{\widetilde{y}}} \sum_{i:y_i=y} \sum_{\widetilde{i}:y_{\widetilde{i}}=\widetilde{y}} \|\boldsymbol{x}_i - \boldsymbol{x}_{\widetilde{i}}\|,$$



(a) $\{\boldsymbol{x}_i\}_{i=1}^{n}$ with $(n_1, n_2) = (90, 10)$ (b) $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ with $(n'_1, n'_2) = (10, 90)$

FIGURE 33.14

Example of class-balance weighted LS. The test class priors are estimated as $\widehat{p}'(y = 1) = 0.18$ and $\widehat{p}'(y = 2) = 0.82$, which are used as weights in class-balance weighted LS.

$$\widehat{b}_y = \frac{1}{n' n_y} \sum_{i'=1}^{n'} \sum_{i:y_i=y} \|\boldsymbol{x}'_{i'} - \boldsymbol{x}_i\|.$$

A MATLAB code for class-balance weighted LS is provided in Fig. 33.13, and its behavior is illustrated in Fig. 33.14. This shows that the class prior can be estimated reasonably well and class-balance weighted learning contributes to improving the classification accuracy for test input points $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$.