Notes on Statistical and Machine Learning

Kernel Smoothing Methods

Chapter: *9* **Prepared by:** *Chenxi Zhou*

This note is prepared based on *Chapter 6, Kernel Smoothing Methods* in Hastie, Tibshirani, and Friedman (2009).

I. One-Dimensional Kernel Smoothers

1. **Introduction to Kernel Smoothing:** By kernel smoothing, we fit a different but simple model separately at each query point $\mathbf{x}_0 \in \mathbb{R}^p$ using only observations close to the target point $\mathbf{x}_0$.

2. **Setup:** Throughout this section, we assume that the predictor lies on the real line, i.e., $x \in \mathbb{R}$. Let $\{(x_i, y_i)\}_{i=1}^n$ be a set of training data points, and $x_0 \in \mathbb{R}$ be the target point.

3. **Review of k-nearest Neighbor Regression:** Recall that the k-nearest neighbor average is

$$\hat{f}(x_0) = \text{Ave}\big(y_i \,|\, x_i \in \mathcal{N}_k(x_0)\big), \tag{1}$$

which is an estimate of the regression function $\mathbb{E}(Y \,|\, X = x_0)$. In (1), $\mathcal{N}_k(x_0)$ is the set of k points nearest to x in the Euclidean distance. However, this estimate is discontinuous in x_0.

4. **Motivation of Kernel Smoothing Methods:** The general idea of the kernel smoothing method to estimate $\mathbb{E}(Y \,|\, X = x_0)$ is, similar to the k-nearest neighbor, to use the observations close to x_0. The localization is achieved via a weighting function or a kernel function $K_\lambda(x_0, x_i)$, which assigns a weight to x_i based on its distance to x_0.

   The choice of the weighting function or the kernel function and its bandwidth (or window size) parameter $\lambda > 0$ controls the width of the neighborhood and the degree of the smoothness of the estimate of the regression function.

5. **Nadaraya-Watson Kernel-Weighted Average:** With a choice of the kernel function of the form

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right), \tag{2}$$

the *Nadaraya-Watson kernel-weighted average* is

$$\hat{f}_0(x_0) = \frac{\sum_{i=1}^n K_\lambda(x_0, x_i) y_i}{\sum_{j=1}^n K_\lambda(x_0, x_j)} = \sum_{i=1}^n \frac{K_\lambda(x_0, x_i)}{\sum_{j=1}^n K_\lambda(x_0, x_i)} y_i. \tag{3}$$

From the second equality, it is obvious that the weight associated with the i-th observation is

$$w_i(x_0) = \frac{K_\lambda(x_0, x_i)}{\sum_{j=1}^n K_\lambda(x_0, x_j)}, \qquad \text{for all } i = 1, \cdots, n.$$

In (2), $D : \mathbb{R} \to [0, \infty)$ is any smooth function such that $D(x) \geq 0$ and satisfies

$$\int_{\mathbb{R}} D(x)\mathrm{d}x = 1, \qquad \int_{\mathbb{R}} xD(x)\mathrm{d}x = 0, \qquad \text{and} \qquad \int_{\mathbb{R}} x^2 D(x)\mathrm{d}x < \infty.$$

Some choices of the kernel function D are:

(a) Epanechnikov quadratic kernel:

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2), & \text{if } |t| \leq 1; \\ 0, & \text{otherwise.} \end{cases}$$

(b) Tri-cube kernel:

$$D(t) = \begin{cases} \left(1 - |t|^3\right)^3, & \text{if } |t| \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

(c) Gaussian kernel:

$$D(t) = \exp\left(-\frac{t^2}{2}\right).$$

Remark 1. The Nadaraya-Watson kernel-weighted average (3) can be characterized as the solution to the following minimization problem

$$\underset{\alpha(x_0)}{\text{minimize}} \left\{ \sum_{i=1}^n K_\lambda(x_0, x_i)\big(y_i - \alpha(x_0)\big)^2 \right\}.$$

Remark 2. The kernel function can be easily generalized to higher dimensions.

Remark 3. In (1), each of the k nearest points is given the equal weight of $1/k$. In Nadaraya-Watson kernel-weighted average, we assign weights that die off smoothly with distance from the target point x_0.

Remark 4. The fitted function (3) is continuous and smooth.

6. **More General Kernel Width:** Note that in (2), we use a fixed constant bandwidth parameter. One can choose a more general function for the kernel width, for example,

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right),$$

where $h_\lambda(x_0)$ is a width function that determines the width of the neighborhood at the specific x_0.

Remark 1. In (1), the neighborhood size k replaces λ, and we have $h_k(x_0) = |x_0 - x_{[k]}|$, where $x_{[k]}$ is the k-th closest x_i to x_0.

Remark 2. In (2), we have $h_\lambda(x) = \lambda$ for all $x \in \mathbb{R}$.

7. Practical Considerations in Applications:

- The smoothing parameter λ determines the width of the local neighborhood. Large λ (averaging more observations) implies lower variance but higher bias, and small λ (averaging less observations) implies higher variance but lower bias.
- Issues can arise with k-nearest neighbor when there are ties in the $x_i's$. The kernel method is less concerned with this issue.
- In the Nadaraya-Waston kernel-weighted average, the boundary issue arises. It is typical that the region closer to the boundaries contains fewer observations and the kernel is asymmetric in these boundary regions. This leads to the *bias* in the Nadaraya-Waston kernel-weighted estimates.
- If the data in the interior of the domain are *not* equally spaced, the Nadaraya-Waston kernel-weighted average can also contain a large bias.

8. Local Linear Regression: *Locally weighted regression* solves a separate weighted least squares problem at each target point x_0

$$\underset{\alpha(x_0),\beta(x_0)}{\text{minimize}} \left\{ \sum_{i=1}^{n} K_\lambda(x_0, x_i) \Big(y_i - \alpha(x_0) - \beta(x_0) x_i \Big)^2 \right\}. \tag{4}$$

The resulting locally linear regression estimate is

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0) x_0,$$

where $(\hat{\alpha}(x_0), \hat{\beta}(x_0))$ is the solution to (4).

We could write $\hat{f}(x_0)$ in the vector-matrix notation as follows:

$$\hat{f}(x_0) = \mathbf{b}(x_0)^\top (\mathbf{B}^\top \mathbf{W}(x_0) \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{W}(x_0) \mathbf{Y} = \sum_{i=1}^{n} l_i(x_0) y_i \tag{5}$$

where $\mathbf{b}(x) = (1, x)^\top$ is a vector-valued function, each row of the matrix $\mathbf{B} \in \mathbb{R}^{n \times 2}$ is $\mathbf{b}(x_i)^\top$, and the matrix $\mathbf{W}(x_0) \in \mathbb{R}^{n \times n}$ is the diagonal matrix with the i-th diagonal element being $K_\lambda(x_0, x_i)$.

Remark 1. Even though we use the entire data set to estimate $\alpha(x_0)$ and $\beta(x_0)$, we *only* evaluate the fitted value at $x = x_0$.

Remark 2. Note from (5) that the fitted value at $x = x_0$ is a linear combination of y_i's with the weights $l_i(x_0)$. There weights $l_i(x_0)$'s are referred to as *equivalent kernel*.

Remark 3. Local linear regression automatically modifies the kernel to correct the bias exactly to first order, a phenomenon known as *automatic kernel carpentry*. More precisely, consider the following series expansion of the true regression f around x_0

$$\begin{aligned}\mathbb{E}\big[\hat{f}(x_0)\big] &= \sum_{i=1}^{n} l_i(x_0) f(x_i) \\ &= f(x_0)\sum_{i=1}^{n} l_i(x_0) + f'(x_0)\sum_{i=1}^{n}(x_i - x_0) l_i(x_0) \\ &\quad + \frac{f''(x_0)}{2}\sum_{i=1}^{n}(x_i - x_0)^2 l_i(x_0) + R,\end{aligned}$$

where the remainder term R depends only on the third- and higher-order derivatives of f, and is small under suitable smoothness assumptions. Using the definition of $l_i(x_0)$'s, we have

$$\sum_{i=1}^{n} l_i(x_0) = 1 \tag{6}$$

and

$$\sum_{i=1}^{n}(x_i - x_0) l_i(x_0) = 0. \tag{7}$$

Therefore, the bias $\mathbb{E}[\hat{f}(x_0)] - f(x_0)$ only depends on *quadratic* and *high-order* terms in the expansion of f.

We show (6) and (7) hold. By matrix algebra, we have

$$\mathbf{B}^\top \mathbf{W}(x_0)\mathbf{B} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

where

$$A_{11} = \sum_{i=1}^{n} K_\lambda(x_i, x_0), \quad A_{12} = A_{21} = \sum_{i=1}^{n} K_\lambda(x_i, x_0) x_i, \quad A_{22} = \sum_{i=1}^{n} K_\lambda(x_i, x_0) x_i^2$$

and, hence,

$$(\mathbf{B}^\top \mathbf{W}(x_0)\mathbf{B})^{-1} = \frac{1}{A_{11}A_{22} - A_{12}^2} \begin{pmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11}. \end{pmatrix}$$

To show (6), we have

$$\begin{aligned}\sum_{i=1}^{n} l_i(x_0) &= \mathbf{b}(x_0)^\top (\mathbf{B}^\top \mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^\top \mathbf{W}(x_0)\mathbf{1}_n \\ &= \begin{pmatrix}1\\ x_0\end{pmatrix}^\top \frac{1}{A_{11}A_{22}-A_{12}^2}\begin{pmatrix}A_{22} & -A_{12}\\ -A_{21} & A_{11}\end{pmatrix}\begin{pmatrix}A_{11}\\ A_{12}\end{pmatrix} \\ &= \frac{1}{A_{11}A_{22}-A_{12}^2}\begin{pmatrix}1\\ x_0\end{pmatrix}^\top \begin{pmatrix}A_{22}A_{11}-A_{12}^2\\ -A_{21}A_{11}+A_{11}A_{12}\end{pmatrix} \\ &= \begin{pmatrix}1\\ x_0\end{pmatrix}^\top \begin{pmatrix}1\\ 0\end{pmatrix} \\ &= 1.\end{aligned}$$

To show (7), we let $\mathbf{x} := (x_1, \cdots, x_n)^\top \in \mathbb{R}^n$ and have

$$\begin{aligned}\sum_{i=1}^{n} l_i(x_0)(x_i - x_0) &= \mathbf{b}(x_0)^\top (\mathbf{B}^\top \mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^\top \mathbf{W}(x_0)(\mathbf{x} - x_0\mathbf{1}_n) \\ &= \begin{pmatrix}1\\ x_0\end{pmatrix}^\top \frac{1}{A_{11}A_{22}-A_{12}^2}\begin{pmatrix}A_{22} & -A_{12}\\ -A_{21} & A_{11}\end{pmatrix}\begin{pmatrix}A_{12}-x_0A_{11}\\ A_{22}-x_0A_{12}\end{pmatrix} \\ &= \begin{pmatrix}1\\ x_0\end{pmatrix}^\top \frac{1}{A_{11}A_{22}-A_{12}^2}\begin{pmatrix}-x_0(A_{11}A_{22}-A_{12}^2)\\ A_{11}A_{22}-A_{21}^2\end{pmatrix} \\ &= \begin{pmatrix}1\\ x_0\end{pmatrix}^\top \begin{pmatrix}-x_0\\ 1\end{pmatrix} \\ &= 0.\end{aligned}$$

9. **Local Polynomial Regression:** We extend the idea above from the linear model to the polynomial model and consider the following optimization problem

$$\min_{\alpha(x_0),\beta_1(x_0),\cdots,\beta_d(x_0)} \left\{\sum_{i=1}^{N} K_\lambda(x_i, x_0)\left(y_i - \alpha(x_0) - \sum_{j=1}^{d}\beta_j(x_0)x_i^j\right)^2\right\},$$

and the fitted value is

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \sum_{j=1}^{d}\hat{\beta}_j(x_0)x_0^j.$$

Remark 1. The bias in the local polynomial regression only depends on the $(d+1)$-th and higher order derivatives of the true regression function.

Remark 2. With the decreasing bias, increasing d leads to a higher variance. Assume the model is of the form $y_i = f(x_i) + \varepsilon_i$, where ε_i's are i.i.d with mean 0 and variance σ^2. Then,

$$\mathrm{Var}\left[\hat{f}(x_0)\right] = \sigma^2 \cdot \|\mathbf{l}(x_0)\|_2^2,$$

where $\mathbf{l}(x_0) \in \mathbb{R}^n$ is the vector of equivalent kernel weights at x_0. In addition, $\|\mathbf{l}(x_0)\|_2$ increases with the polynomial degree d, and therefore, there is bias-variance tradeoff in selecting polynomial degree.

Remark 3. The local polynomial regression yields linear estimators in the sense that the resulting estimate $\hat{f}(x_0)$ can be written as a linear combination of y_i's, for $i = 1, \cdots, n$.

II. Selecting the Width of the Kernel

1. **Effects of the Kernel Width λ:**
   - For the *Epanechnikov* or *tri-cube* kernel with metric width, λ is the radius of the support region;
   - For the *Gaussian* kernel, λ is the standard deviation.
   - In the k-nearest neighbor regression, k, the number of nearest neighbors, plays the role of λ. The larger the value of k is, the more data points we are averaging and the wider the averaging window is.
2. **Bias-Variance Tradeoff in Local Polynomial Regression:** There is a natural *bias-variance tradeoff* as we change the width of the averaging window:
   - If the window is *narrow*, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0. Its variance will be relatively large and the bias will tend to be small;
   - If the window is *wide*, the variance of $\hat{f}(x_0)$ will be small relative to the variance of any y_i, and the bias will be higher.
3. **How to Choose λ:** In practice, one can use the leave-one-out cross-validation, C_p, or k-fold cross-validation to choose the optimal value of λ.

III. Local Regression in $\mathbb{R}^p$

1. **Goal:** We extend the kernel smoothing and local regression to higher dimensions. More precisely,
   (a) the *Nadarava-Watson kernel smoother* fits locally with weights supplied by a p-dimensional kernel;
   (b) local linear regression fits a hyperplane locally in the sample space, by weighted least squares with weights supplied by a p-dimensional kernel.
2. **Notation:** Let $\mathbf{b}(\mathbf{x})$ be a vector of polynomial terms in $\mathbf{x} \in \mathbb{R}^p$ of maximum degree d. For example, with $d = 2$ and $p = 2$, we have $\mathbf{b}(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)^\top \in \mathbb{R}^6$. Let $\{(\mathbf{x}_i^\top, y_i)^\top\}_{i=1}^n$ be the training data, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for all $i = 1, \cdots, n$.

3. **Problem Statement:** To obtain an estimate at each $\mathbf{x}_0 \in \mathbb{R}^p$, we solve the following optimization problem

$$\underset{\boldsymbol{\beta}(\mathbf{x}_0)}{\text{minimize}} \left\{ \sum_{i=1}^{n} K_\lambda(\mathbf{x}_0, \mathbf{x}_i) \big(y_i - b(\mathbf{x}_i)^\top \boldsymbol{\beta}(\mathbf{x}_0)\big)^2 \right\}, \tag{8}$$

and the fitted value at $\mathbf{x} = \mathbf{x}_0$ is

$$\hat{f}(\mathbf{x}_0) = \mathbf{b}(\mathbf{x}_0)^\top \widehat{\boldsymbol{\beta}}(\mathbf{x}_0),$$

where $\widehat{\boldsymbol{\beta}}(\mathbf{x}_0)$ the solution to (8).

4. **Kernel Function in $\mathbb{R}^p$:** Typically, the kernel function in $\mathbb{R}^p$ is a radial function, such as the *radial* Epanechnikov or tri-cube kernel of the form

$$K_\lambda(\mathbf{x}_0, \mathbf{x}) = D\left(\frac{\|\mathbf{x} - \mathbf{x}_0\|_2}{\lambda}\right),$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

Remark. It is suggested to standardize each predictor to unit standard deviation prior to smoothing.

5. **Caveats:** Local polynomial regression becomes less useful in dimensions higher than two or three for the following reasons:

   (a) One manifestation of curse of dimensionality is that the fraction of points close to the boundary increases to 1 as the dimensionality increases. Hence, the boundary effects in two or higher dimensional can be a big problem as the fraction of points on the boundary is large, especially when the boundary is irregular;

   (b) It is impossible maintain low bias (i.e., localness) and low variance (i.e., a sizable sample in the neighborhood) *simultaneously* as the dimension increases, without the total sample size increasing exponentially with p;

   (c) Visualization of $\hat{f}$, which is one of the primary goals of smoothing, in higher dimensions is difficult.

IV. Structured Local Regression Models in $\mathbb{R}^p$

1. **Structured Kernels:** The previous section proposed to use the kernel that is a function of the Euclidean norm of $\mathbf{x} - \mathbf{x}_0$, which puts *equal* weights to each coordinate. One natural modification is to use a positive definite matrix $\mathbf{A}$ to weigh the different coordinates, leading to the following kernel function

$$K_{\lambda,\mathbf{A}}(\mathbf{x}_0, \mathbf{x}) = D\left(\frac{(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}_0)}{\lambda}\right).$$

- If $\mathbf{A}$ is diagonal, one can increase or decrease the influence of X_j by increasing or decreasing $\mathbf{A}_{jj}$;
- If the predictors are highly correlated, one can use the covariance matrix of predictors to tailor $\mathbf{A}$ so that such correlations among predictors can be alleviated.

2. **Structured Regression Functions:** We fit a regression function

$$\mathbb{E}[Y \mid X] = f(X_1, X_2, \cdots, X_p),$$

where $X = (X_1, \cdots, X_p)^\top \in \mathbb{R}^p$ and every level of interaction is potentially present. One approximation is to use ANOVA decomposition of the form

$$f(X_1, X_2, \cdots, X_p) = \alpha + \sum_{j=1}^{p} g_j(X_j) + \sum_{k<l} g_{kl}(X_k, X_l) + \cdots, \tag{9}$$

and then introduce certain structures by eliminating some of the higher-order terms.

- *Additive Model:* Additive model only assume *main effect* terms exist, i.e.,

$$f(X) = \alpha + \sum_{j=1}^{p} g_j(X_j).$$

If we assume that all but the k-th term is known, then one can estimate g_k by local regression of $Y - \sum_{j\neq k} g_j(X_j)$ on X_k and continue this process for each coordinate until convergence.

- *Varying Coefficient Model:* Assume that we divide the predictor vector $X \in \mathbb{R}^p$ into a set $(X_1, \cdots, X_q)^\top$ with $q < p$ and a set of the remaining predictors collectively, denoted by $Z \in \mathbb{R}^{p-q}$. We assume the conditionally linear model

$$f(X) = \alpha(Z) + \beta_1(Z)X_1 + \beta_2(Z)X_2 + \cdots + \beta_q(Z)X_q.$$

Note that, for a given Z, the preceding equation is linear in $X_1, \cdots, X_q$ but each coefficient can vary with Z.

We can fit a model by locally weighted least squares formulated as

$$\underset{\alpha(\mathbf{z}_0), \boldsymbol{\beta}(\mathbf{z}_0)}{\text{minimize}} \left\{ \sum_{i=1}^{n} K_\lambda(\mathbf{z}_0, \mathbf{z}) \Big(y_i - \alpha(\mathbf{z}_0) - x_{1i}\beta_1(\mathbf{z}_0) - \cdots - x_{qi}\beta_q(\mathbf{z}_0) \Big)^2 \right\}.$$

V. Local Likelihood and Other Methods

1. **Motivation:** The concept of local regression and varying coefficient models is extremely *broad*: any parametric model can be made *local* if the fitting method accommodates observation weights.

2. **Local Likelihood Models:** Suppose each observation y_i is associated with $\theta_i := \theta(\mathbf{x}_i) = \mathbf{x}_i^\top \boldsymbol{\beta}$ and we want to estimate $\boldsymbol{\beta}$. We can model $\theta(\,\cdot\,)$ in a more flexible manner using the likelihood function local to $\mathbf{x}_0$ for the inference of $\theta(\mathbf{x}_0) = \mathbf{x}_0^\top \boldsymbol{\beta}(\mathbf{x}_0)$

$$\ell(\boldsymbol{\beta}(\mathbf{x}_0)) := \sum_{i=1}^{n} K_\lambda(\mathbf{x}_i, \mathbf{x}_0) l(y_i, \mathbf{x}_i^\top \boldsymbol{\beta}(\mathbf{x}_0)),$$

and solve the optimization problem

$$\underset{\boldsymbol{\beta}(\mathbf{x}_0)}{\text{maximize}} \Big\{ \ell(\boldsymbol{\beta}(\mathbf{x}_0)) \Big\}.$$

Local likelihood model allows a relaxation from a *global* linear model to one that is *locally* linear.

3. **Local-Likelihood Varying-Coefficient Models:** We consider

$$l(\theta(\mathbf{z}_0)) = \sum_{i=1}^{n} K_\lambda(\mathbf{z}_i, \mathbf{z}_0) \cdot l(y_i, \eta(\mathbf{x}_i, \theta(\mathbf{z}_0))).$$

This will fit a varying coefficient model $\theta(\mathbf{z})$ by maximizing the local likelihood.

4. **Autoregressive Time Series Models:** Consider the autoregressive time series model of order k of the following form

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_k y_{t-k} + \varepsilon_t.$$

Denoting the *lag set* by

$$\mathbf{x}_t := (y_{t-1}, y_{t-2}, \cdots, y_{t-k})^\top,$$

the model looks like a standard linear model $y_t = \mathbf{x}_t^\top \boldsymbol{\beta} + \varepsilon_t$, and is typically fit by the *least squares* method.

One can also fit the time series model by *local* least squares with a kernel $K(\mathbf{x}_0, \mathbf{x}_t)$ allowing the model to vary according to the *short-term history* of the series.

5. **Local Likelihood Regression — Multi-class Linear Logistic Regression Model:** The data set consists of features $\mathbf{x}_i \in \mathbb{R}^p$ and an associated categorical response variable $g_i \in \{1, \cdots, W\}$ and the linear model is of the form

$$\mathbb{P}(G = w \,|\, X = \mathbf{x}) = \frac{\exp(\beta_{w0} + \boldsymbol{\beta}_w^\top \mathbf{x})}{1 + \sum_{u=1}^{W-1} \exp(\beta_{u0} + \boldsymbol{\beta}_u^\top \mathbf{x})},$$

where we set $\beta_{W0} = 0$ and $\boldsymbol{\beta}_W = \mathbf{0}_p$ in the model. The local log-likelihood for this classification problem is

$$\sum_{i=1}^{n} K_\lambda(\mathbf{x}_0, \mathbf{x}_i) \Big\{ \beta_{g_i 0}(\mathbf{x}_0) + \boldsymbol{\beta}_{g_i}(\mathbf{x}_0)^\top (\mathbf{x}_i - \mathbf{x}_0) - \log \Big[1 + \sum_{w=1}^{W-1} \exp(\beta_{w0} + \boldsymbol{\beta}_w(\mathbf{x}_0)^\top (\mathbf{x}_i - \mathbf{x}_0)) \Big] . \Big\}$$

Notice that in the local log-likelihood equation above, we center at $\mathbf{x} = \mathbf{x}_0$.

VI. Kernel Density Estimation and Classification

1. **Problem Statement:** Suppose X is a random variable over $\mathbb{R}$ or a subset of $\mathbb{R}$ whose probability density function is f_X. Suppose we have a random sample $x_1, \cdots, x_n$ drawn from f_X. We wish to estimate the value of f_X at a point x_0.

2. **A First Attempt:** A natural local estimate of density has the following form

$$\hat{f}_X(x_0) = \frac{\big|\{x_i \,|\, x_i \in \mathcal{N}(x_0)\}\big|}{n\lambda},$$

where $\mathcal{N}(x_0)$ is a small metric neighborhood around x_0 of width λ. The *problem* of this estimate is that it is bumpy and *not* smooth.

3. **Smooth Density Estimate:** A smooth estimate is *Parzen estimate* of the form

$$\hat{f}_X(x_0) = \frac{1}{n\lambda}\sum_{i=1}^{n} K_\lambda(x_i, x_0), \tag{10}$$

where a popular choice of K_λ is the Gaussian kernel

$$K_\lambda(x, y) = \phi_\lambda(x - y) = \exp\left(-\frac{(x-y)^2}{2\lambda^2}\right),$$

where $\phi_\lambda(z) = \exp(-z^2/(2\lambda^2))$ is the probability density function of a normal distribution of mean 0 and variance $\lambda^2 > 0$.

Note that (10) can be written as

$$\hat{f}_X(x_0) = \frac{1}{n\lambda}\sum_{i=1}^{n} K_\lambda(x_i, x_0) = \frac{1}{n}\sum_{i=1}^{n} \phi_\lambda(x_0 - x_i) = (\widehat{F}_n * \phi_\lambda)(x_0),$$

which is the convolution of the empirical cumulative distribution function $\widehat{F}_n$ and the Gaussian density ϕ_λ. Here, the empirical cdf $\widehat{F}_n$ puts mass $\frac{1}{n}$ at each observation X_i and is jumpy. To obtain $\hat{f}_X$, we smooth $\widehat{F}_n$ by adding independent Gaussian noise to each observation x_i.

Extensions to Higher-Dimension: We can extend the density estimation idea to $\mathbb{R}^p$ and the result is

$$\hat{f}_X(x_0) = \frac{1}{n(2\lambda^2\pi)^{\frac{p}{2}}}\sum_{i=1}^{n} \exp\left(-\frac{1}{2\lambda^2}\|x_i - x_0\|_2^2\right).$$

4. **Kernel Density Classification:** We use the nonparametric density estimates directly to solve the classification problem. Suppose we have W classes in total. Given the training data set, we fit nonparametric class-conditional density estimates $\hat{f}_w$ for each

of $w = 1, \cdots, W$ separately, and also estimate the class prior probabilities $\hat{\pi}_j$. Then, by Bayes' rule, we obtain the posterior probabilities

$$\widehat{\mathbb{P}}(G = w \,|\, X = \mathbf{x}_0) = \frac{\hat{\pi}_w \hat{f}_w(\mathbf{x}_0)}{\sum_{u=1}^{W} \hat{\pi}_u \hat{f}_u(\mathbf{x}_0)}.$$

Remarks.

(a) If classification is the ultimate goal, learning the separate class densities may *not* be necessary.

(b) When the dimensionality of feature space is large, this approach is not attractive due to difficulties in density estimation.

5. The Naive Bayes Classifier: The naive Bayes classifier applies when the dimensionality p of the feature space is high, making the (joint) density estimation unattractive.

Assumption: Given a class $G = w$, the features $X = (X_1, \cdots, X_p)^\top$ are *independent*, that is,

$$f_w(\mathbf{x}) = \prod_{j=1}^{p} f_{wj}(x_j).$$

Notice that this assumption can reduce the problem dramatically for the following reasons:

- The individual class-conditional marginal densities f_{wj} can each be estimated separately using one-dimensional kernel density estimates, avoiding the difficulties appearing in high-dimensional kernel density estimation;
- If a component of $\mathbf{x}$ is discrete, an appropriate histogram estimate can be used.

Derivation: We use Class W as the base class and consider Class $w = 1, \cdots, W-1$:

$$\begin{aligned}
\log \frac{\mathbb{P}(G = w \,|\, X = \mathbf{x})}{\mathbb{P}(G = W \,|\, X = \mathbf{x})} &= \log \frac{\pi_w f_w(\mathbf{x})}{\pi_W f_W(\mathbf{x})} \\
&= \log \frac{\pi_w \prod_{j=1}^{p} f_{wj}(x_j)}{\pi_W \prod_{j=1}^{p} f_{Wj}(x_j)} \\
&= \log \frac{\pi_w}{\pi_W} + \log \frac{\prod_{j=1}^{p} f_{wj}(x_j)}{\prod_{j=1}^{p} f_{Wj}(x_j)} \\
&= \log \frac{\pi_w}{\pi_W} + \sum_{j=1}^{p} \log \frac{f_{wj}(x_j)}{f_{Wj}(x_j)} \\
&= \alpha_w + \sum_{j=1}^{p} g_{wj}(x_j), \qquad (11)
\end{aligned}$$

where $\alpha_w = \log(\pi_w/\pi_W)$ and $g_{wj} = \log(f_{wj}(x_j)/f_{Wj}(x_j))$. Note that (11) is of the form of a generalized additive model.

Explanation of why Naive Bayes classifier works: Under the independence assumption, the individual class-conditional density estimates may be biased, but this bias does *not* hurt the posterior probabilities as much, especially near the decision regions.

VII. Radial Basis Functions and Kernels

1. **Introduction:** *Kernel methods* achieve flexibility by fitting simple models in a region local to the target point $\mathbf{x}_0$. Localization is achieved via a weighting kernel K_λ, and individual observations receive weights $K_\lambda(\mathbf{x}_0, \mathbf{x}_i)$.

   Radial basis functions combine these ideas, by treating the kernel functions $K_\lambda(\boldsymbol{\xi}, \mathbf{x})$ as basis functions. This leads to the model

$$f(\mathbf{x}) = \sum_{j=1}^{M} \beta_j K_{\lambda_j}(\boldsymbol{\xi}_j, \mathbf{x}) = \sum_{j=1}^{M} \beta_j D\left(\frac{\|\boldsymbol{\xi}_j - \mathbf{x}\|}{\lambda_j}\right),$$

   where each basis element is indexed by a *location* or *prototype* parameter $\boldsymbol{\xi}_j$ and a *scale* parameter λ_j. One popular choice of the kernel function D is the standard Gaussian density function.

2. **Parameter Estimation:**

   (a) *Method 1 — Least Squares Estimation:* We use least squares method for regression to estimate the parameter values $\{\lambda_j, \boldsymbol{\xi}_j, \beta_j\}_{j=1}^M$. We optimize the sum-of-squares with respect to all parameters

$$\underset{\{\lambda_j, \boldsymbol{\xi}_j, \beta_j\}_{j=1}^M}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{M} \beta_j \exp\left(-\frac{(\mathbf{x}_i - \boldsymbol{\xi}_j)^\top (\mathbf{x}_i - \boldsymbol{\xi}_j)}{\lambda_j^2} \right) \right)^2 \right\}. \tag{12}$$

   This model is referred to as a radial basis function (RBF) network.

   The objective function in (12) is *non-convex* and have multiple local minima, and can be solved by using algorithms in neural networks.

   (b) *Method 2 — Two-stage Estimation:* Estimate $\{\lambda_j, \boldsymbol{\xi}_j\}_{j=1}^M$ separately from $\boldsymbol{\beta} := (\beta_0, \beta_1, \cdots, \beta_M)$.

   i. Given $\{\lambda_j, \boldsymbol{\xi}_j\}_{j=1}^M$, estimating $\boldsymbol{\beta}$ is a *least squares* problem;

   ii. Given $\boldsymbol{\beta}$, we can estimate $\{\lambda_j, \boldsymbol{\xi}_j\}_{j=1}^M$ by the following approaches:

   - Fit a Gaussian mixture density model to the feature variables in the training data, i.e., $\mathbf{x}_1, \cdots, \mathbf{x}_n$, and from here we can obtain both $\{\lambda_j, \boldsymbol{\xi}_j\}_{m=1}^M$;
   - Use clustering methods to locate the prototypes $\boldsymbol{\xi}_j$ and let $\lambda_j = \lambda$ as a hyper-parameter.

   (c) *Problem of Treating $\lambda_j = \lambda$:*

   - Assuming $\lambda_j = \lambda$ for all j reduces the parameter set and results in a simpler model to fit;

- This approach creates *holes*, i.e., the regions of $\mathbb{R}^p$ where none of the kernels has appreciable support.

A remedy is to use *renormalized radial basis functions* defined as

$$h_j(x) = \frac{D(\|\mathbf{x} - \boldsymbol{\xi}_j\|_2/\lambda)}{\sum_{k=1}^M D(\|\mathbf{x} - \boldsymbol{\xi}_k\|_2/\lambda)}.$$

Remark. Recall that the Nadaraya-Watson kernel-weighted average in $\mathbb{R}^p$ is of the form

$$\hat{f}(\mathbf{x}_0) = \frac{\sum_{i=1}^n K_\lambda(\mathbf{x}_i, \mathbf{x}_0) y_i}{\sum_{i=1}^n K_\lambda(\mathbf{x}_i, \mathbf{x}_0)} = \sum_{i=1}^n y_i \frac{K_\lambda(\mathbf{x}_i, \mathbf{x}_0)}{\sum_{i=1}^n K_\lambda(\mathbf{x}_i, \mathbf{x}_0)} = \sum_{i=1}^n y_i h_i(\mathbf{x}_0),$$

which can be viewed as an expansion in renormalized radial basis functions. In such an expansion, $\boldsymbol{\xi}_i = \mathbf{x}_i$ and $\hat{\beta}_i = y_i$ for all $i = 1, \cdots, n$.

VIII. Mixture Models for Density Estimation and Classification

1. **Gaussian Mixture Model:** The *Gaussian mixture model* of M components has the form

$$f(x) = \sum_{m=1}^M \alpha_m \phi(\mathbf{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{13}$$

where α_m's are the mixing weights satisfying $\alpha_m \geq 0$ and $\sum_{m=1}^M \alpha_m = 1$ and $\phi(\,\cdot\,; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density function with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. We can use the Gaussian mixture model (13) to do density estimation.

2. **Special Cases of Gaussian Mixture Model:** We consider the following two special cases of the Gaussian mixture model:

   - If the covariance matrices are a multiple of identity matrix, i.e.,

   $$\boldsymbol{\Sigma}_m = \sigma_m \cdot \mathbf{I},$$

   then (13) has the form of a radial basis expansion;
   - If, additionally, $\sigma_m = \sigma > 0$ is fixed, and $M \nearrow n$, the maximum likelihood estimate for (13) approaches the kernel density estimate

   $$\hat{f}_X(\mathbf{x}_0) = \frac{1}{n\lambda} \sum_{i=1}^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i),$$

   where $\hat{\alpha}_m = \frac{1}{n}$ and $\hat{\boldsymbol{\mu}}_m = \mathbf{x}_m$.

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Vol. 1. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.