STAT 534

Lecture 6

Kernel Density Estimation, Regression and Classification

April 18, 2019

mmp@stat.washington.edu

Scribes: Ruojin He*, Yikun Zhang*

1 Introduction

In this lecture, we will discuss Nearest-Neighbor predictors and Kernel Density Estimation approaches on classification and regression problem.

1.1 Classification and Regression Problem

The classification and regression problems fall in a general category of machine learning tasks, the so-called supervised learning. Given a random pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, the basic goal in supervised learning is to construct a prediction function f such that $Y = f(X)$ based on some training data. We often call X the *input*, *predictor*, *feature*, *independent variable*, etc., and Y the *output*, *response*, *dependent variable*, etc. Typically, the input space of X, i.e., $\mathcal{X}$, would simply be $\mathbb{R}^d$ and the training data comprises some (i.i.d.) samples from (X, Y), $\mathcal{D} = \{(x^i, y^i) \in \mathbb{R}^d \times \mathcal{Y}, i = 1, ..., N\}$.
The difference between classification and regression problems emerges on the type of output variables.

- **Classification**: The output Y is *qualitative* and assumes values in a finite set. For instance, $\mathcal{Y} = \{-1, 1\}$.
- **Regression**: The output Y is a *quantitative* measurement, i.e., $\mathcal{Y} \subset \mathbb{R}$.

In a nutshell, the distinction in output type has led to a naming convention for the prediction tasks: *regression* when we predict quantitative outputs, and *classification* when we predict qualitative outputs. These two tasks have a lot in common, and in particular both can be viewed as a task in function approximation.[1]

1.2 Memory-Based Learning

The main topics of today's lecture, nearest-neighbor predictors and kernel density estimation methods, can be classified as examples of *memory-based learning*

*Department of Statistics, University of Washington

(sometimes called *instance-based learning*) within the realm of machine learning. **Memory-based learning** is based on the assumption that in learning a cognitive task from experience people do not extract rules or other abstract representations from their experience, but reuse their memory of that experience directly.[2] In other words, it compares new data instances with those seen in the training data, which have been stored in memory. Since it constructs hypotheses directly from the training instances themselves, the hypothesis complexity can grow with the data[3]: in the worst case, a hypothesis is a list of n training items and the computational complexity of classifying a single new instance is $\mathcal{O}(n)$.

1.3 Nonparametric Model

On the other hand, both Nearest-Neighbor predictors and kernel density estimation approaches are some vivid examplifications of nonparametric models, whose structure is not specified a priori but learned from data. This means that these statistical models are infinite-dimensional, in the sense that the number and nature of the parameters are grown with the size of data.

2 Nearest-Neighbor Predictor

Given a training data $\mathcal{D} = \{(x^i, y^i) \in \mathbb{R}^d \times \mathcal{Y}, i = 1, ..., N\}$, the main idea of Nearest-Neighbor predictor is to assign the label or value of a new instance x as follows:

1. Find the example x^i that is nearest to x under a certain distance metric, says Euclidean distance. Mathematically, $i = \underset{j \in \{1,...,N\}}{\arg\min} ||x - x^j||_2$
2. Assign x the label or value y^i.

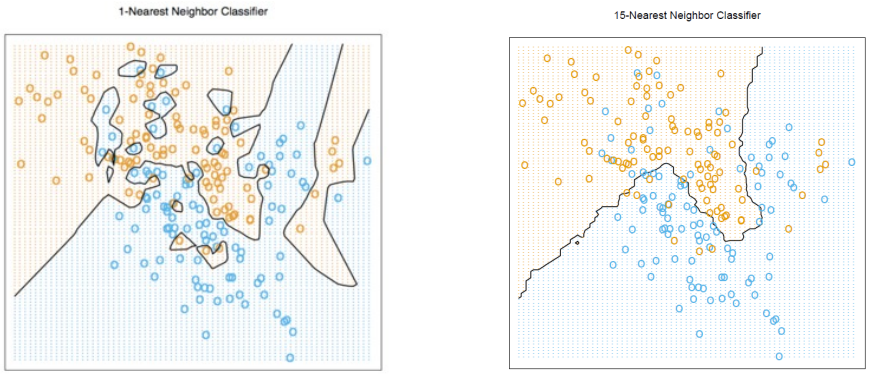
In practice, one uses the K nearest neighbors of x (with $K = 3, 5$ or larger). Then

- **For Classification**: $f(x) =$ the most frequent label among K nearest neighbors (the so-called *Majority Vote*).
- **For Regression**: $f(x) = \frac{1}{K} \sum_{i \text{ neighbor of } x} y^i =$ mean of neighbors' values.

Figure 1 delineates an example when we apply the K-Nearest-Neighbor method on a binary classification problem, where those solid curves indicate the decision boundaries of two classifiers. In a binary classification problem, the **decision boundary** of a classifier is a hypersurface that partitions the underlying vector space into two sets, while the classifier will assign all the points on one side of the decision boundary to one class and all those on the other side to the other class. Upon the decision boundary, the output label of a classifier is ambiguous.[4] As shown by Figure 1, K serves as the smoothing parameter of the K-Nearest-Neighbor classifier, since the decision boundary becomes smoother as the value

of K increases.

Remark. The decision boundary for a K-Nearest-Neighbor classifier is *piecewise linear*.



(a) 1-Nearest-Neighbor (b) 15-Nearest-Neighbor

Figure 1: A Binary Classification Problem Using 1-Nearest-Neighbor and 15-Nearest-Neighbor Classifiers with Decision Boundaries[1]

3 Kernel Density Estimation

Let $x^1, ..., x^N \in \mathbb{R}^d$ be an independent, identically distributed random sample from an unknown distribution P with density function p. Then the **Kernel Density Estimation** can be expressed as

$$\hat{p}_n(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} b\left(\frac{x - x^i}{h}\right),$$

where $b : \mathbb{R}^d \to \mathbb{R}$ is a smooth function such that $b(x) \geq 0$ and[5]

$$\int b(x)dx = 1, \int x \cdot b(x)dx = 0, \text{ and } \sigma_b^2 \equiv \int x^2 \cdot b(x)dx > 0,$$

and $h > 0$ is the bandwidth parameter that controls the amount of smoothing. b is called a kernel function. Two common examples of $b(x)$ are[6]

$$\text{(Gaussian kernel) } b(x) = \frac{\exp(-\frac{||x||^2}{2})}{v_{1,d}}, \; v_{1,d} = \int \exp\left(-\frac{||x||^2}{2}\right) dx,$$

$$\text{(Spherical Kernel) } b(x) = \frac{I(||x|| \leq 1)}{v_{2,d}}, \; v_{2,d} = \int I(||x|| \leq 1)dx.$$

Optional requirements on b are: (i) symmetric with respect to x and x^i, that is, $b(x, x^i) = b(x^i, x) = b\left(\frac{x-x^i}{h}\right)$; (ii) Radial symmetry; (iii) bounded support; (iv) higher order smoothness conditions, etc.

As with kernel regression, the choice of kernel b is not crucial, but the choice of bandwidth h is important.[5] Figure 2 shows density estimates with several different bandwidths using a part of the NACC (National Alzheimers Coordinating Center) Uniform Dataset, version 3.0 (March 2015).[6]

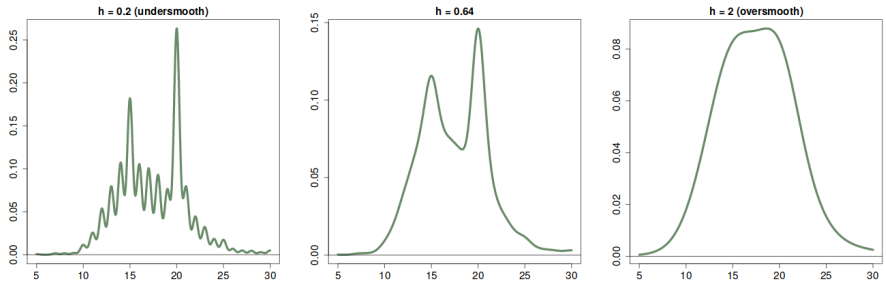


Figure 2: Kernel Density Estimation on Part of NACC Data. Left panel: undersmoothed. Middle panel: just right (bandwidth chosen by the default rule in R)[†]. Right panel: oversmoothed.

Roughly speaking, the computational cost of kernel density estimation is $\mathcal{O}(Nd)$, since we need to calculate the (Euclidean) distance of two d-dimensional data points. In Section 5 and subsequent lectures, we will introduce several efficient neighbor searching methods when the number of data points N is large.

4 Kernel Regression and Classification

4.1 Kernel Regression

In any kernel regression setting, the conditional expectation of the response Y relative to the input X can be written as

$$f(X) = \mathbb{E}(Y|X).$$

As for a training data with i.i.d. samples $\mathcal{D} = \{(x^i, y^i) \in \mathbb{R}^d \times \mathbb{R}, i = 1, ..., N\}$, we can always write

$$y^i = f(x^i) + \epsilon_i, \quad i = 1, ..., n,$$

where $\epsilon_i, i = 1, ..., n$ are i.i.d. random errors with mean zero. Like the K-Nearest-Neighbor approach, the kernel regression interpolate the value of a new data instance based on the neighbor value of this instance, but in a more "smoothed" way. Suppose that $b_h(x, x^i) = b\left(\frac{x-x^i}{h}\right)$, where b is a kernel func-

[†]For instance, see https://stat.ethz.ch/R-manual/R-devel/library/stats/html/bandwidth.html for details

tion defined in Section 3. Then the **kernel regressor** can be written as

$$\hat{f}(x) = \sum_{i=1}^{N} \beta_i(x) \cdot b_h(x, x^i) \cdot y^i,$$

where β_i's are coefficients. If we estimate f as a locally weighted average, i.e., $\sum_{i=1}^{N} \beta_i b_h(x, x^i) = 1$, then $\beta_i = \frac{1}{\sum_{j=1}^{N} b_h(x, x^j)}, j = 1, ..., N$ and the estimator becomes

$$\hat{f}(x) = \sum_{i=1}^{N} \frac{b_h(x, x^i)}{\sum_{j=1}^{N} b_h(x, x^j)} \cdot y^i,$$

which is the well-known **Nataraya-Watson kernel estimator**.[5] In this estimator (or regressor), $\hat{f}(x)$ is always a convex combination of y^i's and the weights are proportional to $b_h(x, x^i)$.

Remark. The Nataraya-Watson estimator is biased if the density of X varies around x.

To alleviate the biased problems for kernel estimators, one can resort to a generalization of kernel regression called *local linear regression*. The procedure goes as follows.

1. Given a *query point* x, compute the weight function $w_i = b_h(x, x^i)$ for all $i = 1, ..., N$.
2. Solve the **weighted sums of squares** $\min_{\beta, \beta_0} \sum_{i=1}^{N} w_i (y^i - \beta^T x^i - \beta_0)^2$ to obtain β, β_0 (β, β_0 depend on x through w_i).[‡]
3. Calculate $\hat{f}(x) = \beta^T x + \beta_0$.

Remark. The Nataraya-Watson estimator solves a local linear regression with fixed $\beta = 0$.

4.2 Kernel (Binary) Classification

Given the training data $\mathcal{D} = \{(x^i, y^i) \in \mathbb{R}^d \times \{Class1, Class2\}, i = 1, ..., N\}$, we can encode the binary responses y^i's into $\{-1, 1\}$. Then the **kernel binary classifier** can be written as

$$\hat{f}(x) \propto \sum_{i=1}^{N} y^i \cdot b\left(\frac{x - x^i}{h}\right),$$

where b is a kernel function defined in Section 3. The prediction is based on the sign of $\hat{f}(x)$, i.e.,

$$y(x) = \begin{cases} 1 & \text{(or Class 1) if } \hat{f}(x) > 0, \\ -1 & \text{(or Class 2) if } \hat{f}(x) < 0. \end{cases}$$

[‡]See section 5.4 in [5] for detailed calculation.

The plug-in estimator $\{x \in \mathcal{R}^d : \hat{f}(x) = 0\}$ of the **solution manifold** $\{x \in \mathcal{R}^d : f(x) = 0\}$ is the decision boundary of the kernel classifier. In addition, the decision boundary for a kernel binary classifier can be viewed as a plug-in estimator of the density level-set at level 0, for which asymptotic properties and visualization techniques have been analyzed and proposed. See [7] for details.

5 KD-Trees & Ball-Tree

Both K-Nearest-Neighbor and kernel prediction methods involve scanning the whole dataset for every single prediction. Given training data $\mathcal{D} = \{(x^i, y^i) \in \mathbb{R}^d \times \mathcal{Y}, i = 1, ..., N\}$,

- for K-Nearest Neighbor, predicting $f(x)$ for a single new instance x involves computing N distances in a d-dimensional space. The computational cost is approximately $\mathcal{O}(Nd)$.
- for kernel methods, finding the data points within the support of the kernel function, usually a d-dim ball with radius r, also involves computing the distances of a single x to all training data points.

The neighbor search (or computing pairwise distances of training data points) is a polynomial-time process but still *computational expensive*, especially when the dimension d is high. Can we design some algorithms that are more efficient? The answer is **Yes**, if we index (i.e. preprocess) the training data. Indexing here means organizing the data in a way that makes finding the neighbors of any given point fast. In particular, with indexing, searching neighbors of a given point x *does not require comparing* x *with all* N *data points*.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Element of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. Springer Series in Statistics. Springer-Verlag New York, 2009.

[2] Walter Daelemans and Antal van den Bosch. *Memory-Based Language Processing*. New York, NY, USA: Cambridge University Press, 2009.

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Third Edition. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[4] Wikipedia Contributors. *Decision boundary*. [Online; Accessed 20-April-2019]. 2018. URL: https://en.wikipedia.org/wiki/Decision_boundary.

[5] Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[6] Yen-Chi Chen. "A tutorial on kernel density estimation and recent advances". In: *Biostatistics & Epidemiology* 1.1 (2017), pp. 161–187.

[7] Yen-Chi Chen, Christopher R. Genovese, and Larry Wasserman. "Density Level Sets: Asymptotics, Inference, and Visualization". In: *Journal of the American Statistical Association* 112.520 (2017), pp. 1684–1696. eprint: https://doi.org/10.1080/01621459.2016.1228536.