CS 5785 – Applied Machine Learning – Lec. 9

Prof. Nathan Kallus, Cornell Tech
Scribes: Yiren Zhou, Christophe Rimann, Yuemin Niu, Wen Guo

October 9, 2018

1 Recap

Inference is essentially opening up the supervised learning "Black box" in an attempt to understand how $\hat{f}$ depends on data. In the previous lecture, we discussed how inference works for the linear model, as well as confidence intervals for coefficients (i.e. how far we are from where we would be with infinite data).

Now, we briefly discuss an aside: standardization and regularization. Recall that the goal of ridge regression is to minimize

$$||Y - X\beta||_2^2 + \lambda \sum_{j=1}^{p} (\beta_j)^2$$

If X and Y have different units, the results will not make sense. As such, to make the penalty fair, often we first standardize the data as follows:

$$x_{ij} = \frac{x_{ij} - u_j}{\sigma_j}$$

We do the same for Lasso, given Lasso's dependence on ridge.
This is not an issue for the following:
- Subset selection
- If features are of similar nature e.g. Pixel brightness; word counts

2 Density Estimation

Consider data

$$Y_1, Y_2 ... Y_n \in \mathcal{R}$$

drawn from a distribution with CDF F and PDF f = F'. How do we understand this distribution from data alone? In some sense, how do we estimate F and F? Estimating F is easy: we can just find the empirical CDF:

$$\hat{F}_n(y) = \frac{1}{n} \sum_{i=1}^{n} (\prod (Y_i <= y))$$

Estimating f is more difficult. We cannot simply differentiate $\hat{F}$, because it is a stepwise function, which would mean we would have infinite slope at the points where the stepwise function is active, and 0 otherwise. In particular, given $y \notin Y_1, Y_2 ... Y_n$, we are not sure how dense Y is at y.

3 Histogram

One approach is to use histograms. Given a list of cutoffs

$$y_1 < y_2 < ... < y_n$$

we can get bins

$$[y_1, y_2), [y_2, y_3), ..., [y_{n-1}, y_n)$$

From there, we can count how much data is in each bin.

$$n_j = \sum_{i=1}^{n} (\prod [Y_i \in [y_{j-1}, y_j]])$$

A histogram is usually just a bar chart with these heights - each bin becomes a bar, with the height being the amount of data in the bin. How do we choose the bins? Usually segment $[\min Y_i, \max Y_i]$ equally. More sophisticated ways that choose bins automatically are implemented in packages; we will not be covering them in depth.

To make it a density estimate, we need to normalize, so we define a j s.t. $y \in [y_{j-1}, y_j)$

$$\hat{f}_n(y) = \frac{n_j}{n} \frac{1}{y_j - y_{j-1}}$$

There's a problem with this: histograms are not smooth, and don't really look like f?

4 Kernel Density Estimation

Density estimation in general is a key aspect of some supervised learning algorithms we'll be using later on. For now, we are going to use kernel density estimation (a.k.a Parzen Window density estimation) to help solve some of the issues with Histograms, namely their lack of smoothness. Kernel density estimation is essentially a "smoother" version of a histogram (i.e. a continuous version of a histogram). Analogously, while histograms required us to specify bin widths, we now have to specify a kernel width.
This is represented as the following equation:

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

ithin this equation, $K_\lambda(x_0, x) = \phi(|x - x_0|/\lambda)$ represents a kernel of width λ, with center at x_0 for a point x. λ is correspondingly the "smoothing parameter". We can choose different kernels K (as long as they are non negative functions); one such example and the one used in class is $K_\lambda = \varphi(\frac{u}{\lambda}$, where φ is the PDF of the standard Gaussian/Normal distribution: $\varphi(u) = \frac{1}{(2\pi)^{\frac{1}{2}}} e^{-\frac{1}{2}u^2}$. The problem

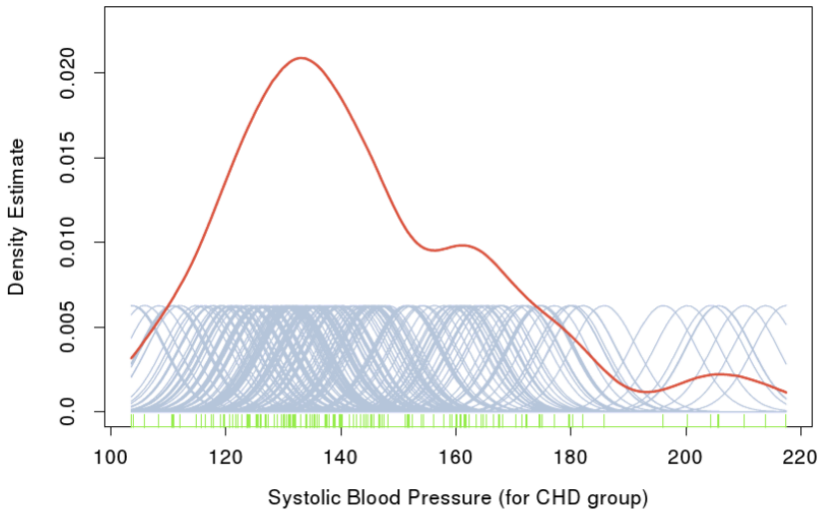

Figure 1: Systolic blood pressures

with the Gaussian as shown is that it does not integrate to 1; as such, we divide by $\frac{1}{n\lambda}$ to accommodate for this in the equation for $\hat{f}_X(x_0)$ above.

We can look to figure **??** for an example. In this figure, we can see the green ticks representing different levels of systolic blood pressure, where each tick represents one measurement. The blue lines are the kernels formed around each tick: as we can see, we are adding one normal distribution centered around each green tick. The red line is the resulting kernel density estimate, which is formed by summing the kernels at every given point. As such, where the Gaussian functions are more crowded, the kernel density estimate will be larger, while where the Gaussians are more sparse, the kernel density estimate is smaller.

In figure **??**, we can see the effect of changing λs on the resulting KDE. The red line represents a KDE with λ=0.05, black is 0.337, and green is 2. The grey is the actual true density (the standard normal distribution). We can see that smaller values are rougher, while larger values are smoother. Somewhere in the middle, the grey is fairly close to our true density. In order to establish the correct value for λ, we will use Cross Validation, with log likelhiood as our loss function.

5 Kernel Regression

Kernel regression comes from the idea of using weighted average of training data to make prediction. It is actually very similar to k-nearest neighbors regression. However, the exponential weights allow kernel regression to give a continuous and smooth prediction. We can also get the equivalent formula from a kernel density estimation (KDE) perspective. Figure **??** shows a comparison between KNN and kernel regression. The KNN prediction is stepwise and discontinous. We can see that kernel regression gives a smooth prediction, which makes it

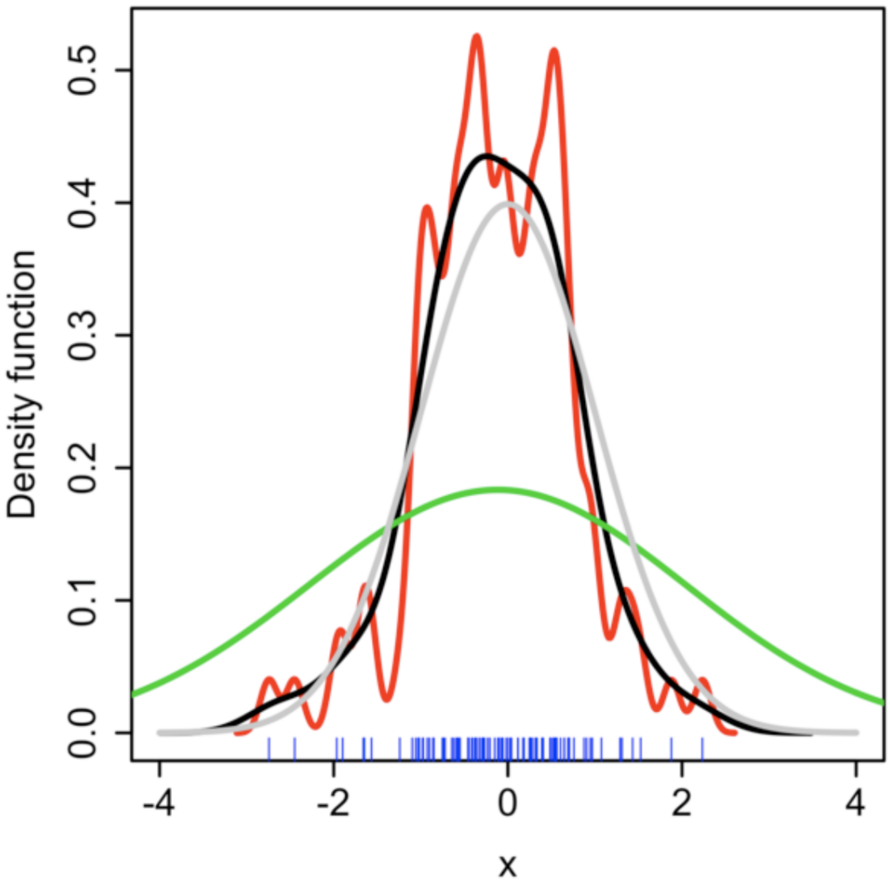

Figure 2: Changing λ values for KDE

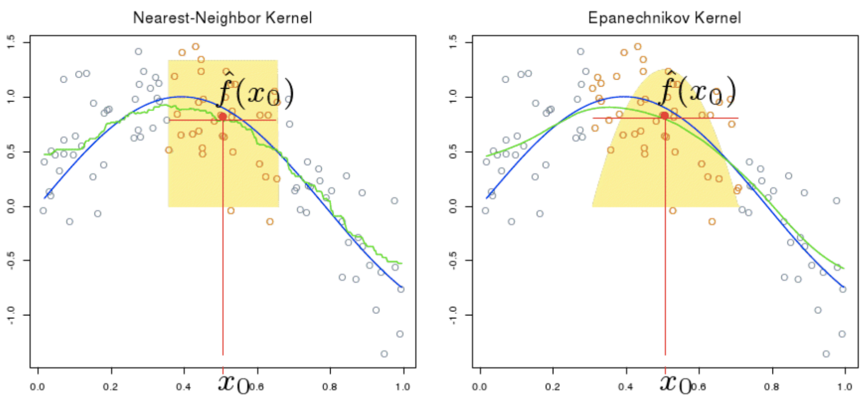

Figure 3: KNN and Kernel Regression

more favorable.

For a regression problem, assume the distribution of Y and X are known. The best prediction for $Y|X = x$ is the conditional expectation:

$$\begin{aligned} E[Y|X = x] &= \int y f_{Y|X=x}(y)dy \\ &= \int y \frac{f_{Y,X}(y,x)}{f_X(x)} dy \\ &= \frac{\int y f_{Y,X}(y,x)dy}{\int f_{Y,X}(y,x)dy} \end{aligned}$$

However, the joint distribution of Y and X, $f_{Y,X}(y,x)$, is unknown and we need to estimate it. If substituted with a kernel density estimation (KDE), we get the kernel regression formulation:

$$E[Y|X = x] \approx \frac{\int y \hat{f}_{Y,X}(y,x)dy}{\int \hat{f}_{Y,X}(y,x)dy}$$

where $\hat{f}_{Y,X}(y,x)$ is substituted by the kernel density estimator for the joint distribution of (Y,X):

$$\hat{f}_{Y,X}(y,x) = \frac{1}{n\lambda} \sum_{i=1}^{n} K_\lambda \left(\begin{pmatrix} y \\ x \end{pmatrix} - \begin{pmatrix} Y_i \\ X_i \end{pmatrix} \right)$$

Then the kernel regression estimator $\hat{\mu}_n(x)$ becomes:

$$\begin{aligned}
\hat{\mu}_n(x) &= \frac{\frac{1}{n\lambda}\sum_{i=1}^n \int y\phi\left(\frac{(y-Y_i)^2+\|x-X_i\|_2^2}{\lambda}\right)}{\frac{1}{n\lambda}\sum_{i=1}^n \int \phi\left(\frac{(y-Y_i)^2+\|x-X_i\|_2^2}{\lambda}\right)} \\
&= \frac{\sum_{i=1}^n \int ye^{-\frac{1}{2\lambda^2}(y-Y_i)^2} e^{-\frac{1}{2\lambda^2}\|x-X_i\|^2} dy}{\sum_{i=1}^n \int e^{-\frac{1}{2\lambda^2}(y-Y_i)^2} e^{-\frac{1}{2\lambda^2}\|x-X_i\|^2} dy} \\
&= \frac{\sum_{i=1}^n K_\lambda(x-X_i)\int ye^{-\frac{1}{2\lambda^2}(y-Y_i)^2} dy}{\sum_{i=1}^n K_\lambda(x-X_i)\int e^{-\frac{1}{2\lambda^2}(y-Y_i)^2} dy} \\
&= \frac{\sum_{i=1}^n K_\lambda(x-X_i)\int (u+Y_i)e^{-\frac{1}{2\lambda^2}u^2} du}{\sum_{i=1}^n K_\lambda(x-X_i)\int e^{-\frac{1}{2\lambda^2}u^2} du} \\
&= \frac{\sum_{i=1}^n K_\lambda(x-X_i)Y_i}{\sum_{i=1}^n K_\lambda(x-X_i)}
\end{aligned}$$

We can see from the formula that kernel regression is actually a weighted average for all training points, where every point (Y_i, X_i), $i = 1, ..., n$ is given weight $\frac{K_\lambda(x-X_i)}{\sum_{i=1}^n K_\lambda(x-X_i)}$.

Figure **??** shows the comparison between different scale parameter λ. We can see that λ acts like the number of neighbors k in KNN, which controls the bias-variance tradeoff. A bigger λ puts more weight on local data, and a smaller λ smoothes over the whole domain. Figure **??** plots seven different kernels. They all integrate to 1 and are valid kernels for KDE. The Gaussian kernel is the most commonly used one among them.

6 Kernel Classification

We can use KDE to obtain classification as well. For instance, let Y be +1 or -1. Then,

$$P(Y=1|X=x) \geq 0.5 \Longleftrightarrow E[Y|X=x] \geq 0$$

$$E[Y|X=x] = E[(\prod[Y=1] - \prod[Y \neq 1])|X=x]$$

From here, we can apply the kernel regression to estimate E[Y—X=x] and apply this estimate.

$$\hat{\mu}(x) \geq 0 \Longleftrightarrow \frac{1}{n\lambda}\sum_{i=1}^n K_\lambda(x-x_i)y_i \geq 0$$

$$\frac{1}{n\lambda}\sum_{Y_i=+1} K_\lambda(x-x_i) \geq \frac{1}{n\lambda}\sum_{Y_i=-1} K_\lambda(x-x_i)$$

$$\frac{n_1}{n}\frac{1}{n_1\lambda}\sum_{Y_i=+1} K_\lambda(x-x_i) \geq \frac{n_{-1}}{n}\frac{1}{n_{-1}\lambda}\sum_{Y_i=-1} K_\lambda(x-x_i)$$

where n_1 is the number of positive examples. Applying the KDE of x among positive examples $\hat{f}$, we get:

$$\hat{\pi}_1\hat{f}_1(x) \geq \hat{\pi}_{-1}\hat{f}_{-1}(x)$$

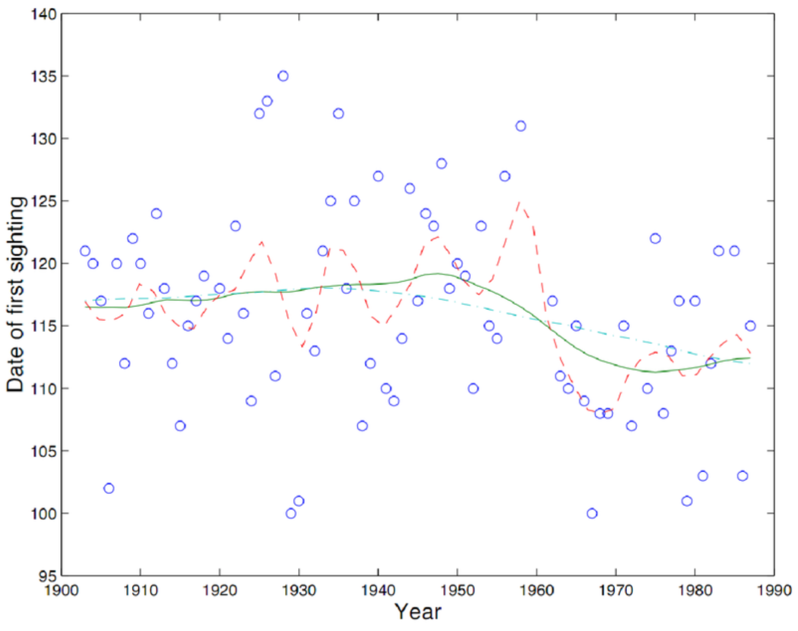

Figure 4: Kernel Regression with Different Scale λ

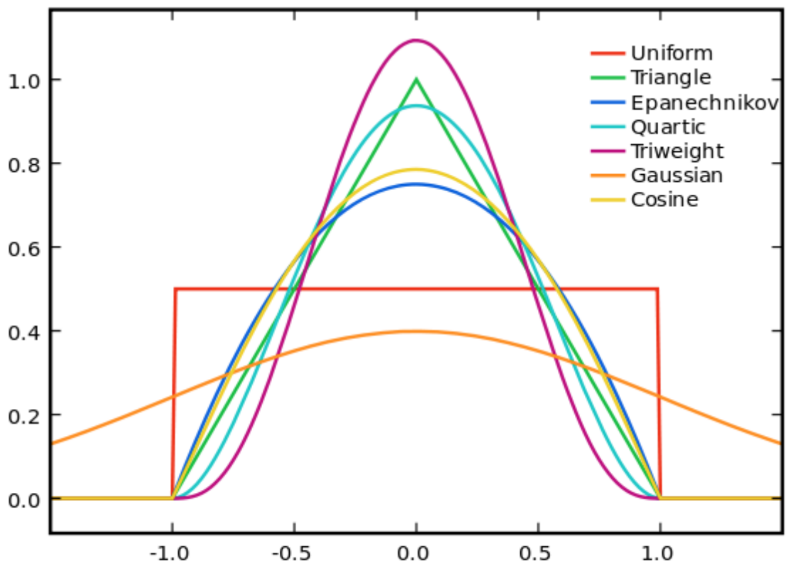

Figure 5: Different Kernels for KDE

By rearranging terms and apply logs, we can obtain the kernel estimated log odds:

$$log\frac{\hat{\pi}_1}{\hat{\pi}_{-1}} + \frac{log\hat{f}_1(x)}{log\hat{f}_{-1}(x)} \geq 0$$

Rather than just allowing Y to take on values of + or - 1, we can generalize this classification beyond two possible values with the following:

$$\hat{P}(Y = j|X = x) = \frac{\hat{\pi}_j \hat{f}_j(x)}{\sum_{k=1}^{m} \hat{\pi}_k \hat{f}_k(x)}$$

We can see that as dimensions grow bigger, equivalent sized boxes cover less data.