Directed Reading Program Final write up

Gefei Shen

May 2024

This semester we are focusing on exploring the different concepts in statistics including kernel density estimator, causal inference, double machine learning, variational autoencoder (VAE), diffusion model and data thinning. I would focus more on the kernel density estimator (KDE) during this write up.

1 KDE

1.1 Definition

Kernel Density Estimator is a non-parametric method used to estimate the probability density function of a random variable. It works by placing a kernel function at each data point and then summing these functions to create a smooth estimate of the overall data distribution.

$$\begin{aligned}\hat{f}_h : x \mapsto& \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)\\ &= \frac{1}{n}\sum_{i=1}^{n} K_h(X_i - x),\end{aligned}$$

where $K_h : u \mapsto \frac{1}{h}K(u/h)$.

Notice: X_i is the only randomness here, x is fixed, h is bandwidth and K is kernel function.

1.2 Rate of convergence

It is really interesting to talk about the rate of convergence when we talk about KDE. Therefore, I will give some computational proof. Firstly, we decompose the mean square error into bias square and variance (Bias-Variance Trade-off):

$$\begin{aligned}\mathrm{MSE}(\hat{f}_n(x)) &= \mathbb{E}[(\hat{f}_n(x) - f(x))^2]\\ \mathrm{MSE}(\hat{f}_n(x)) &= \mathrm{Bias}^2(\hat{f}_n(x)) + \mathrm{Var}(\hat{f}_n(x))\end{aligned}$$

1.2.1 Bias

$$\begin{aligned}
\mathbb{E}[\hat{f}_h(x_0)] - f(x_0) &= \frac{1}{nh}\sum_{i=1}^{n} \mathbb{E}\left[K\left(\frac{X_i - x_0}{h}\right)\right] - f(x_0) \\
&= \frac{1}{h}\mathbb{E}\left[K\left(\frac{X_1 - x_0}{h}\right)\right] - f(x_0) \\
&= \frac{1}{h}\int K\left(\frac{X_i - x_0}{h}\right) f(x)\,dx - f(x_0) \\
&= \frac{1}{h}\int K(u) f(x_0 + uh)\,du - f(x_0) \quad (\text{using } u = \frac{x_i - x_0}{h}) \\
&= \int K(u)[f(x_0 + uh) - f(x_0)]\,du \\
&= \int K(u)(f'(\tilde{x}_{uh}) - f'(x_0))uh + K(u)f'(x_0)uh\,du \\
&= \int K(u)(f'(\tilde{x}_{uh}) - f'(x_0))uh\,du \\
|Bias| &= |\int K(u)(f'(\tilde{x}_{uh}) - f'(x_0))uh\,du| \\
&\leq \int |K(u)(c)uh|\,du \\
&= \int K(u)h|u||f'(\tilde{x}_{uh}) - f'(x_0)|\,du \\
&\leq \int K(u)h|u||\tilde{x}_{uh} - x_0|L\,du \\
&\leq h^2 \int K(u)^2 L\,du \\
&= Lh^2\sigma_k^2
\end{aligned}$$

Bias Squared

$$\begin{aligned}
\text{Bias}^2 &\leq L^2 h^4 \sigma_k^4 \\
&= O(h^4)
\end{aligned}$$

1.2.2 Variance

$$\begin{aligned}
\mathrm{Var}(\hat{f}_h(x_0)) &= \mathrm{Var}\left(\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x_0}{h}\right)\right)\\
&= \frac{1}{(nh)^2}\sum_{i=1}^{n}\mathrm{Var}\left(K\left(\frac{X_i - x_0}{h}\right)\right)\\
&= \frac{1}{nh^2}\mathrm{Var}\left(K\left(\frac{X_1 - x_0}{h}\right)\right)\\
&\leq \frac{1}{nh^2}\mathbb{E}\left[K^2\left(\frac{X_1 - x_0}{h}\right)\right]\\
&= \frac{1}{nh^2}\int_{-\infty}^{\infty} K^2\left(\frac{x_1 - x_0}{h}\right) f(x_1)\, dx_1\\
&= \frac{1}{nh^2}\int_{-\infty}^{\infty} K^2(u) f(x_0 + uh) h\, du \quad (\text{using } u = \frac{x_1 - x_0}{h})\\
\text{Let } k_1 &= \inf\{x : K(x) > 0\}\\
\text{Let } k_2 &= \sup\{x : K(x) > 0\}\\
&= \frac{1}{nh}\int_{k_1}^{k_2} K(u)^2 f(uh + x_0)\, du\\
&\leq \frac{1}{nh} S\cdot C\\
&= O\left(\frac{1}{nh}\right)
\end{aligned}$$

1.2.3 Optimal h

$$\begin{aligned}
\mathrm{MSE}(\hat{f}_h(x_0)) &= \mathrm{Bias}^2(\hat{f}_h(x_0)) + \mathrm{Var}(\hat{f}_h(x_0))\\
&= O(h^4) + O\left(\frac{1}{nh}\right)
\end{aligned}$$

I want them to converge at the same rate.

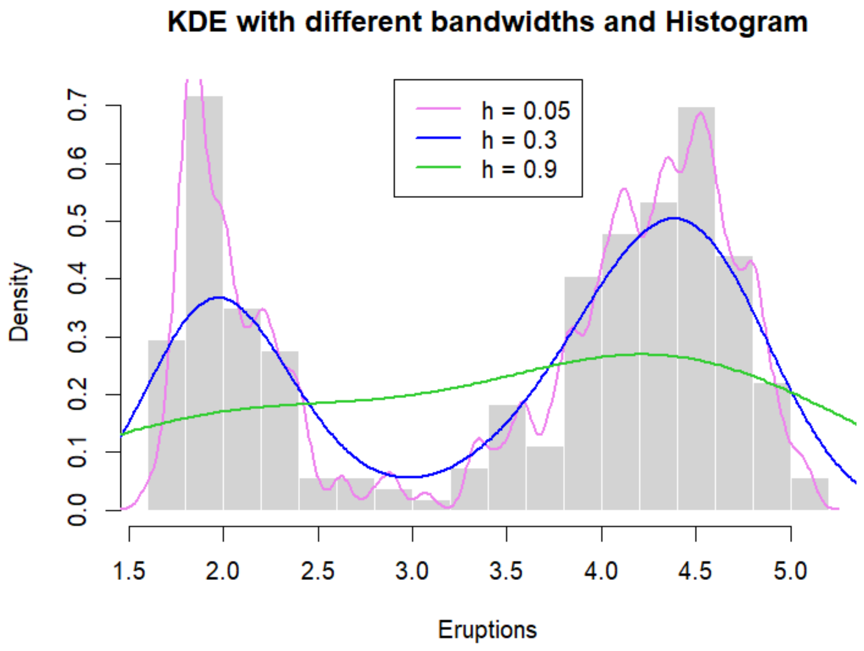
$$\begin{aligned}
h^4 &= \frac{1}{nh}\\
h &= n^{-\frac{1}{5}}
\end{aligned}$$

$$\begin{aligned}
\mathrm{Bias}^2(\hat{f}_h(x_0)) &= O((n^{-1/5})^4) = O(n^{-4/5})\\
\mathrm{Var}(\hat{f}_h(x_0)) &= O\left(\frac{1}{n\cdot n^{-1/5}}\right) = O(n^{-4/5})
\end{aligned}$$

$$\text{MSE}(\hat{f}_h(x_0)) = O(n^{-4/5})$$

This means the convergence rate cannot be lower than $n^{-\frac{4}{5}}$. As the order of the kernel function increases, the mean square error of this kernel density estimator would converge faster.

1.3 Demo



From the plot above, the histogram is the original data, $h = 0.3$ is the optimal h from the previous part, $O(n^{-\frac{1}{5}})$. We could find when h is too large, it would cause underfitting, and when h is too small, it would cause overfitting.

2 Causal inference

During the second and third weeks, we talked about the intuition and some computational proofs about causal inference.

2.1 Some background

Causation requires mechanistic understanding, indicating that intervention in one variable leads to change in another.

One classic example here: Zeus is a patient waiting for a heart transplant. On January 1, he receives a new heart. Five days later, he dies. Imagine that we can somehow know that had Zeus not received a heart transplant on January 1, he would have been alive five days later.

On the other hand, Hera is another patient waiting for a heart transplant. On January 1, she receives a new heart. Five days later she was alive. Imagine we can somehow know that, had Hera not received the heart on January 1, she would still have been alive five days later.

Then we discuss the average treatment effect (ATE) and Randomized Controlled Trials (RCTs).

3 Double Machine Learning

During the fourth week, we discussed double machine learning. We started from linear regression: $Y = E[Y|X] + \epsilon$ and then we talked about $E[Y|X] = \beta x + \alpha$. $\beta = \frac{\text{Cov}(x,y)}{\text{Var}(x)}$. Then what if $Y = \beta x + \gamma z + \epsilon$.

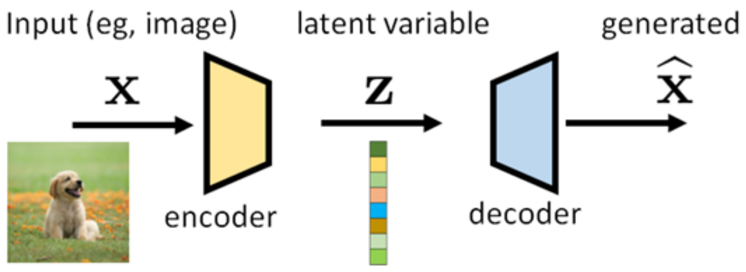
$$\begin{aligned}
&\text{if } Z \perp x, \text{ random variable} \\
&\text{if } Z \not\perp x, \beta = \frac{\text{Cov}(x^{\perp z}, y^{\perp z})}{\text{Var}(x^{\perp z})} \\
&Y^{\perp z} \text{ is the residual of (Y Z)} \\
Y &= \gamma z + \epsilon \\
Y^{\perp z} &= \hat{\epsilon} = (Y - \hat{\gamma} z)
\end{aligned}$$

Then we discussed the convergence probability of double machine learning.

4 Variational Auto Encoder

During the fifth and sixth weeks, we discussed the Variational Auto Encoder (VAE). I would give some KL divergence computational proof and some demo in this report.

4.1 Construction of VAE

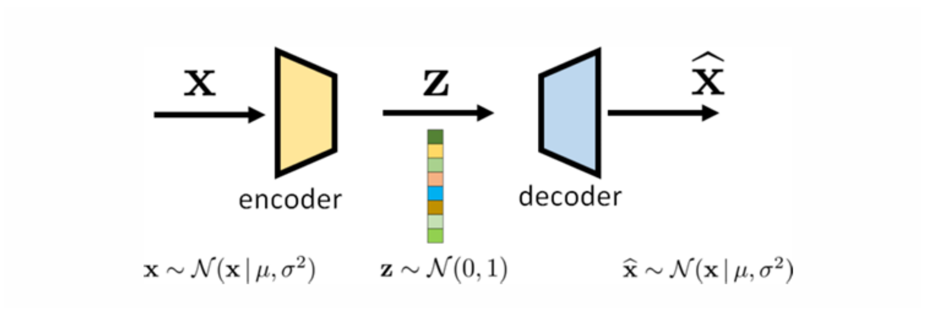


In VAE we would extremely focus on the

$$p(x, z) = p(x|z) \cdot p(z) = p(z|x) \cdot p(x) \tag{1}$$

- $p(x)$ is the distribution of x.
- $p(z)$ is the prior and the distribution of the latent variable. In this analysis, I would let $p(z) = \mathcal{N}(0, I)$.
- $p(z|x)$ is the posterior and is the probability related to the encoder.
- $p(x|z)$ is the probability related to the decoder.

In VAE, we usually give an input x and obtain the output by first encoding it to a latent variable z using the approximate posterior distribution $q(z|x)$, and then decoding z to generate $\hat{x}$ using the generative distribution $p(x|z)$.


X
Z
$\widehat{X}$
encoder
decoder
$\mathbf{x} \sim \mathcal{N}(\mathbf{x} \mid \mu, \sigma^2)$
$\mathbf{z} \sim \mathcal{N}(0, 1)$
$\hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{x} \mid \mu, \sigma^2)$

4.2 ELBO and KL divergence

Then, we discuss the evidence lower bound (ELBO) and KL divergence with some mathematical proof.

$$\text{ELBO}(x) = E_{q_\phi(z|x)}[log_{p\theta}(x|z)] - D_{KL}(q_\phi(z|x))||p(z)$$

4.3 KL divergence is non-negative

We prove that the KL divergence is non-negative.

$$\begin{aligned}
D_{\mathrm{KL}}(p\|q) &= \int p(x)\log\frac{p(x)}{q(x)}\,dx \\
&= \int p(x)\log\frac{p(x)}{q(x)}\,dx \\
&= -\int p(x)\log\frac{q(x)}{p(x)}\,dx \\
&= \int p(x)\log\frac{q(x)}{p(x)}\,dx \\
&\text{By Jensen's Inequality, for the convex function:} \\
E[f(X)] &\leq f(E[X]) \\
E\left[\log\frac{p(x)}{q(x)}\right] &\leq \log E\left[\frac{p(x)}{q(x)}\right] \\
E_p\left[\log\frac{q(x)}{p(x)}\right] &\leq \log E_p\left[\frac{q(x)}{p(x)}\right] \\
&= \log\int p(x)\frac{q(x)}{p(x)}\,dx \\
&= \log\int q(x)\,dx \\
&= \log 1 \\
&= 0 \\
\Rightarrow E_p\left[\log\frac{q(x)}{p(x)}\right] &\leq 0 \\
\int p(x)\log\frac{q(x)}{p(x)}\,dx &\leq 0 \\
-\int p(x)\log\frac{q(x)}{p(x)}\,dx &\geq 0 \\
\int p(x)\log\frac{p(x)}{q(x)}\,dx &\geq 0
\end{aligned}$$

This proves that the KL divergence is non-negative.

5 Diffusion model

During seventh week, we discussed the diffusion model and focus on the score matching.

5.1 Langevin Dynamics

The **Langevin dynamics** for sampling from a known distribution p(x) is an iterative procedure for $t = 1, \dots, T$:

$$x_{t+1} = x_t + \tau \nabla_x logp(x_t) + \sqrt{2\tau} z, z \in Z(0, I)$$

where τ is the step size which user can control, and x_0 is the white noise.

5.2 Score Function

- Score function: $s_x(\theta) = \nabla_\theta logp_\theta(x)$
- Stein's score function $s_\theta(x) = \nabla_x logp_\theta(x)$

5.3 Denoising Score Matching

The **Denoising Score Matching** has a loss function defined as

$$J_{\text{DSM}}(\theta) = E_{p(x)}[\frac{1}{2}||s_\theta(x + \sigma z) + \frac{z}{\sigma^2}||^2] \tag{2}$$

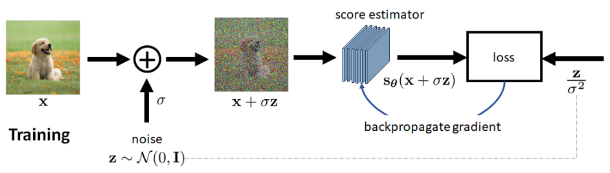

score estimator
loss
$\mathbf{x}$
Training
σ
noise
$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
$\mathbf{x} + \sigma\mathbf{z}$
$\mathbf{s}_\theta(\mathbf{x} + \sigma\mathbf{z})$
backpropagate gradient
$\frac{\mathbf{z}}{\sigma^2}$

Figure 21: Training of $\mathbf{s}_\theta$ for denoising score matching. The network $\mathbf{s}_\theta$ is trained to estimate the noise.

6 Data Thinning

During week 8, we talked about data thining. It is a way to split data to avoid double using the data in some specific circumstances (e.g clustering). Therefore, we use data thining to split the data. First given a X following the distributing F. We split X into X_1 and X_2, where X_1 and X_2 are independent of each other and follows follows same distribution family (e.g Gaussian, Poisson, etc.). Then, $X_1 + X_2 = X$. To be more specific

(i) Let $X^{(1)} \in F_{\epsilon\lambda}$, $X^{(2)} \in F_{(1-\epsilon)\lambda}$, and $X^{(1)} \perp X^{(2)}$

(ii) make x from $X \in F_\lambda$

(iii) Draw $X^{(1)}$ from X. Let $X^{(2)} = X - X^{(1)}$

7 Tree Based Method

Lastly, we discuss tree-based methods. First, we introduce the decision tree, which serves as the foundation for various ensemble techniques. We then explore Bootstrap Aggregating (Bagging), which involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model. Random Forest extends Bagging by constructing multiple decision trees and combining their outputs for more robust predictions, adding an extra layer of randomness by selecting a subset of features for each split in the trees. Finally, we discuss a vertical structured tree-method:XGBoost. Boosting works in a similar way as random forest, except that the trees are grown sequentially: each tree is grown using information from previously grown trees