CSCI567 Machine Learning (Spring 2021)

Sirisha Rambhatla

University of Southern California

March 24, 2021

Outline

1. Logistics
2. Review of last lecture
3. Density estimation

Outline

Logistics

- Today is the tracking day for the project.
- When you submit course deliverables, check them to ensure you have uploaded the correct files.
- In **April 23, 2021**'s lecture we will have **Alice Xiang**, *Senior Research Scientist and AI Ethics Lead at Sony AI* to talk about **Fairness in AI**. If possible, please plan to join the class live!

Outline

General EM algorithm

Step 0 Initialize $\boldsymbol{\theta}^{(1)}$, $t = 1$

Step 1 (E-Step) **update the posterior of latent variables**

$$q_n^{(t)}(\cdot) = p(\cdot \mid \boldsymbol{x}_n \,; \boldsymbol{\theta}^{(t)})$$

and obtain **Expectation** of complete likelihood

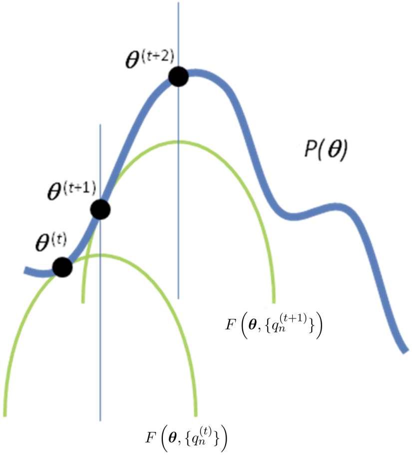
$$Q(\boldsymbol{\theta} \,; \boldsymbol{\theta}^{(t)}) = \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n \,; \boldsymbol{\theta})\right]$$

Step 2 (M-Step) **update the model parameter** via **Maximization**

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \,; \boldsymbol{\theta}^{(t)})$$

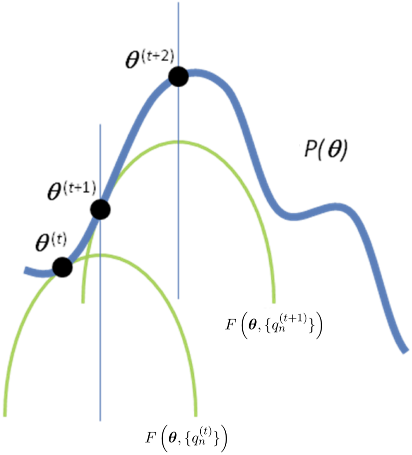
Step 3 $t \leftarrow t + 1$ and return to Step 1 if not converged

Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

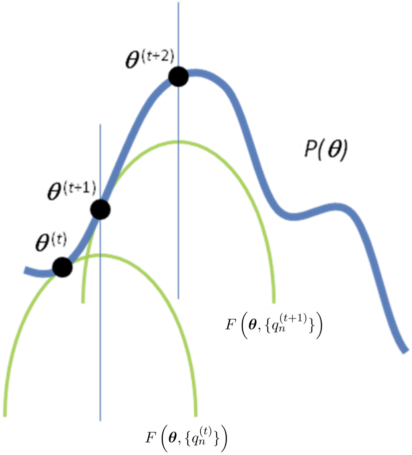
Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$P(\boldsymbol{\theta}^{(t+1)}) \geq F\left(\boldsymbol{\theta}^{(t+1)}; \{q_n^{(t)}\}\right)$$

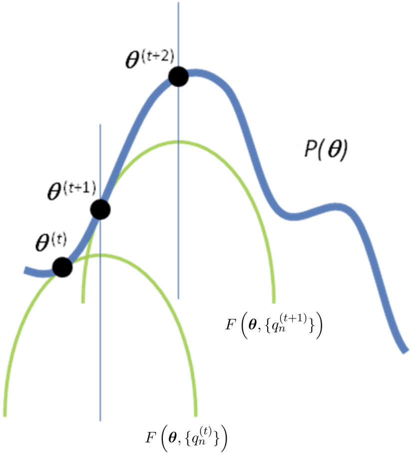
Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$
\begin{aligned}
P(\boldsymbol{\theta}^{(t+1)}) &\geq F\left(\boldsymbol{\theta}^{(t+1)}; \{q_n^{(t)}\}\right) \\
&\geq F\left(\boldsymbol{\theta}^{(t)}; \{q_n^{(t)}\}\right)
\end{aligned}
$$

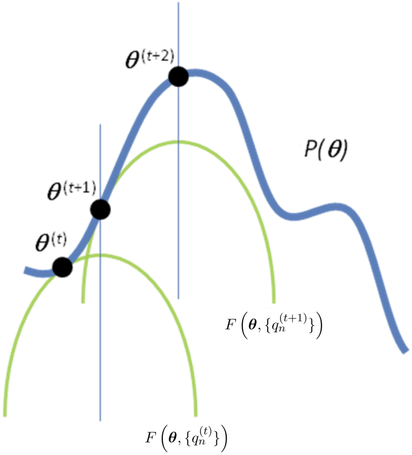
Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$
\begin{aligned}
P(\boldsymbol{\theta}^{(\mathrm{t+1})}) &\geq F\left(\boldsymbol{\theta}^{(\mathrm{t+1})}; \{q_n^{(t)}\}\right) \\
&\geq F\left(\boldsymbol{\theta}^{(\mathrm{t})}; \{q_n^{(t)}\}\right) \\
&= P(\boldsymbol{\theta}^{(\mathrm{t})})
\end{aligned}
$$

Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$\begin{aligned} P(\boldsymbol{\theta}^{(t+1)}) &\geq F\left(\boldsymbol{\theta}^{(t+1)}; \{q_n^{(t)}\}\right) \\ &\geq F\left(\boldsymbol{\theta}^{(t)}; \{q_n^{(t)}\}\right) \\ &= P(\boldsymbol{\theta}^{(t)}) \end{aligned}$$

So EM always increases the objective value and will converge to some local maximum (similar to K-means).

Applying EM to learn GMMs

EM for clustering:

Step 0 Initialize $\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for each $k \in [K]$

Step 1 (E-Step) **update the "soft assignment"** (fixing parameters)

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n) \propto \omega_k N\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$

Step 2 (M-Step) **update the model parameter** (fixing assignments)

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N} \qquad \boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Step 3 return to Step 1 if not converged

Outline

Density estimation

Observe what we have done indirectly for clustering with GMMs:

Density estimation

Observe what we have done indirectly for clustering with GMMs:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_n \overset{i.i.d.}{\sim} p$).

Density estimation

Observe what we have done indirectly for clustering with GMMs:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_n \overset{i.i.d.}{\sim} p$).

This is exactly the problem of *density estimation*, another important unsupervised learning problem.

Density estimation

Observe what we have done indirectly for clustering with GMMs:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_n \overset{i.i.d.}{\sim} p$).

This is exactly the problem of *density estimation*, another important unsupervised learning problem.

Useful for many downstream applications

- we have seen clustering already, will see more today

Density estimation

Observe what we have done indirectly for clustering with GMMs:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_n \stackrel{i.i.d.}{\sim} p$).

This is exactly the problem of *density estimation*, another important unsupervised learning problem.

Useful for many downstream applications

- we have seen clustering already, will see more today
- these applications also *provide a way to measure quality of the density estimator*

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,; \boldsymbol{\theta})$$

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,; \boldsymbol{\theta})$$

Examples:

- GMM: $p(\boldsymbol{x}\,|\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,;\boldsymbol{\theta})$$

Examples:

- GMM: $p(\boldsymbol{x}\,|\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$
- Multinomial: a discrete variable with values in $\{1, 2, \ldots, K\}$ s.t.

$$p(x = k\,;\boldsymbol{\theta}) = \theta_k$$

where $\boldsymbol{\theta}$ is a distribution over K elements.

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,;\boldsymbol{\theta})$$

Examples:

- GMM: $p(\boldsymbol{x}\,|\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$
- Multinomial: a discrete variable with values in $\{1, 2, \ldots, K\}$ s.t.

  $$p(x = k\,;\boldsymbol{\theta}) = \theta_k$$

  where $\boldsymbol{\theta}$ is a distribution over K elements.

Size of $\boldsymbol{\theta}$ is independent of the training set size, so it's parametric.

Parametric methods: estimation

Again, we apply **MLE** to learn the parameters $\boldsymbol{\theta}$:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \ln p(x_n \,; \boldsymbol{\theta})$$

Parametric methods: estimation

Again, we apply **MLE** to learn the parameters $\boldsymbol{\theta}$:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \ln p(x_n \,; \boldsymbol{\theta})$$

For some cases this is intractable and we can use EM to approximately solve MLE (e.g. GMMs).

Parametric methods: estimation

Again, we apply **MLE** to learn the parameters $\boldsymbol{\theta}$:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \ln p(x_n \,; \boldsymbol{\theta})$$

For some cases this is intractable and we can use EM to approximately solve MLE (e.g. GMMs).

For some other cases this admits a simple closed-form solution (e.g. multinomial).

MLE for multinomial

The log-likelihood is

$$\sum_{n=1}^{N} \ln p(x = x_n \,; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \theta_{x_n}$$

MLE for multinomial

The log-likelihood is

$$\sum_{n=1}^{N} \ln p(x = x_n \,; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \theta_{x_n}$$

$$= \sum_{k=1}^{K} \sum_{n:x_n=k} \ln \theta_k$$

MLE for multinomial

The log-likelihood is

$$\begin{aligned}\sum_{n=1}^{N} \ln p(x = x_n \,; \boldsymbol{\theta}) &= \sum_{n=1}^{N} \ln \theta_{x_n} \\ &= \sum_{k=1}^{K} \sum_{n : x_n = k} \ln \theta_k = \sum_{k=1}^{K} z_k \ln \theta_k\end{aligned}$$

where $z_k = |\{n : x_n = k\}|$ is **the number of examples with value k**.

MLE for multinomial

The log-likelihood is

$$\sum_{n=1}^{N} \ln p(x = x_n ; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \theta_{x_n}$$
$$= \sum_{k=1}^{K} \sum_{n:x_n=k} \ln \theta_k = \sum_{k=1}^{K} z_k \ln \theta_k$$

where $z_k = |\{n : x_n = k\}|$ is **the number of examples with value k**.

The solution is simply

$$\theta_k = \frac{z_k}{N} \propto z_k,$$

i.e. the fraction of examples with value k.

Nonparametric methods

Can we estimate *without assuming a fixed generative model?*

Nonparametric methods

Can we estimate *without assuming a fixed generative model?*

Yes, **kernel density estimation (KDE)** is a common approach

Nonparametric methods

Can we estimate *without assuming a fixed generative model?*

Yes, **kernel density estimation (KDE)** is a common approach

- here "kernel" means something different from what we have seen for "kernel function" (in fact it refers to several different things in ML)

Nonparametric methods

Can we estimate *without assuming a fixed generative model?*

Yes, **kernel density estimation (KDE)** is a common approach

- here "kernel" means something different from what we have seen for "kernel function" (in fact it refers to several different things in ML)
- the approach is nonparametric: it keeps the entire training set

Nonparametric methods

Can we estimate *without assuming a fixed generative model?*

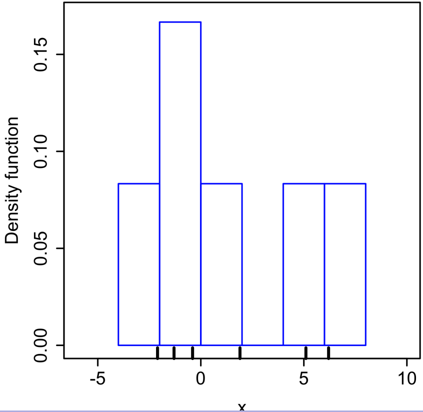
Yes, **kernel density estimation (KDE)** is a common approach

- here "kernel" means something different from what we have seen for "kernel function" (in fact it refers to several different things in ML)
- the approach is nonparametric: it keeps the entire training set
- we focus on the 1D (continuous) case

High level idea

picture from Wikipedia

Construct something similar to a **histogram**:

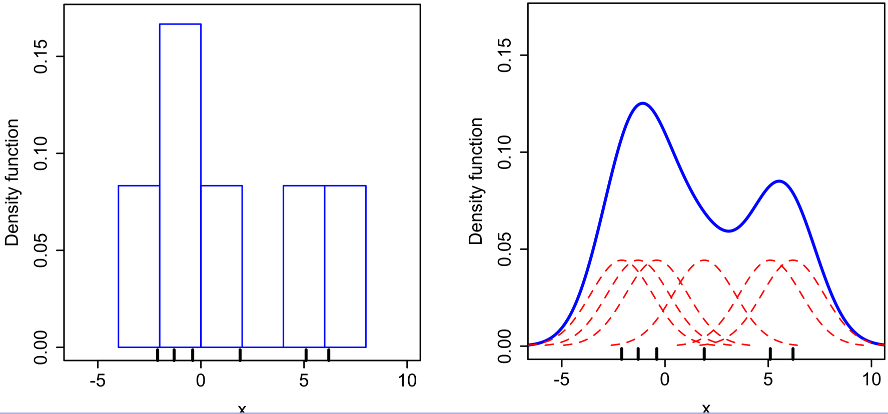


High level idea

picture from Wikipedia

Construct something similar to a **histogram**:

- for each data point, create a "bump" (via a Kernel)

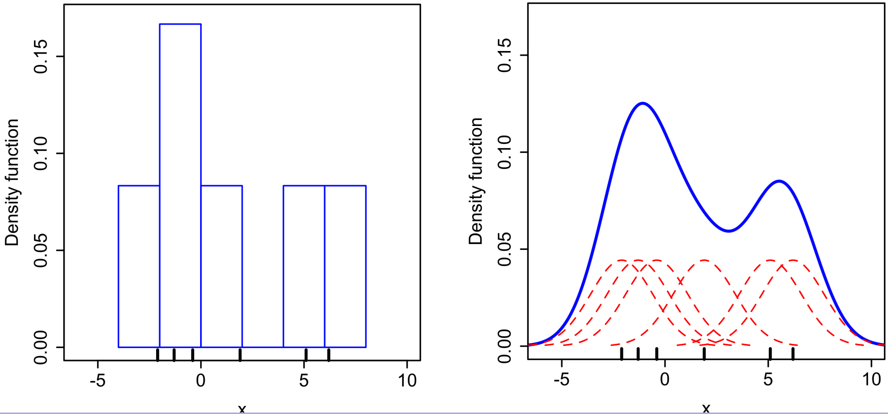


High level idea

picture from Wikipedia

Construct something similar to a **histogram**:

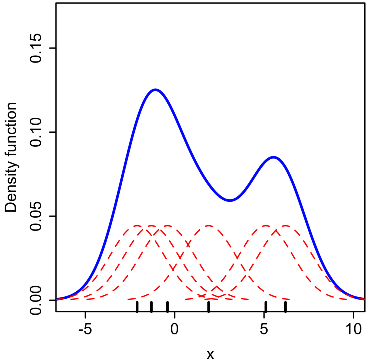
- for each data point, create a "bump" (via a Kernel)
- sum up or average all the bumps



Kernel

KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K\left(x - x_n\right)$$

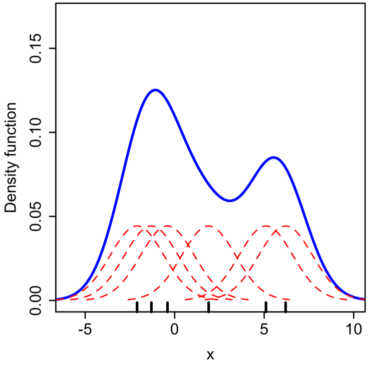


Kernel

KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K\left(x - x_n\right)$$

e.g. $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, the standard Gaussian density



Kernel

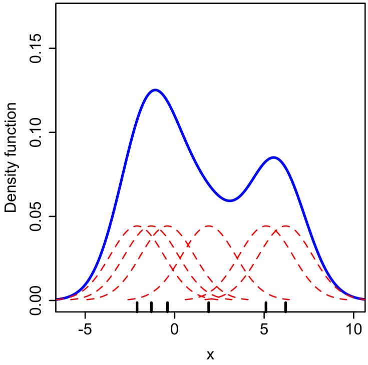
KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K\left(x - x_n\right)$$

e.g. $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, the standard Gaussian density

Kernel needs to satisfy:

- symmetry: $K(u) = K(-u)$



Kernel

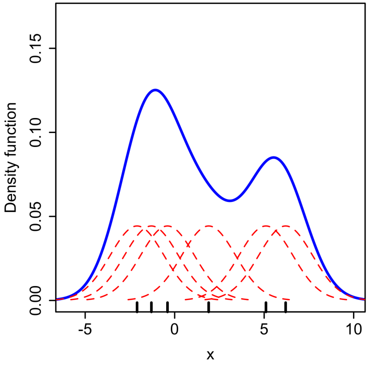
KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K(x - x_n)$$

e.g. $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, the standard Gaussian density

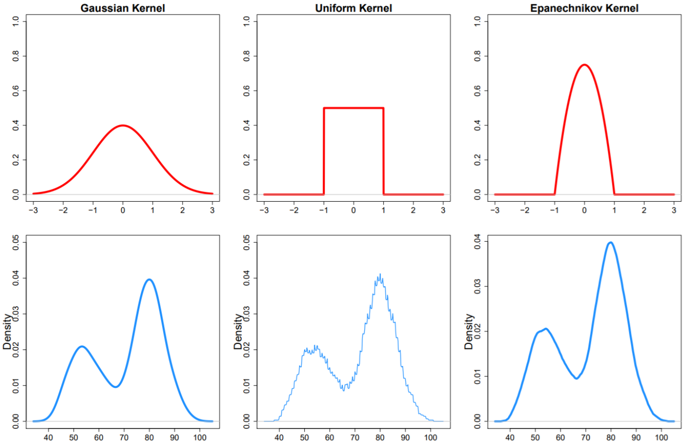
Kernel needs to satisfy:

- symmetry: $K(u) = K(-u)$
- $\int_{-\infty}^{\infty} K(u) du = 1$, makes sure p is a density function.



Different kernels $K(u)$

$$\frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}} \qquad \frac{1}{2}\mathbb{I}[|u| \leq 1] \qquad \frac{3}{4}\max\{1 - x^2, 0\}$$



Bandwidth

If $K(u)$ is a kernel, then for any $h > 0$

$$K_h(u) \triangleq \frac{1}{h} K\left(\frac{u}{h}\right) \qquad \text{(stretching the kernel)}$$

can be used as a kernel too (verify the two properties yourself)

Bandwidth

If $K(u)$ is a kernel, then for any $h > 0$

$$K_h(u) \triangleq \frac{1}{h} K\left(\frac{u}{h}\right)$$

(stretching the kernel)

can be used as a kernel too (verify the two properties yourself)

So general KDE is determined by both the kernel K and the bandwidth h

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K_h\left(x - x_n\right) = \frac{1}{Nh} \sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right)$$

Bandwidth

If $K(u)$ is a kernel, then for any $h > 0$

$$K_h(u) \triangleq \frac{1}{h} K\left(\frac{u}{h}\right) \qquad \text{(stretching the kernel)}$$

can be used as a kernel too (verify the two properties yourself)

So general KDE is determined by both the kernel K and the bandwidth h

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K_h\left(x - x_n\right) = \frac{1}{Nh} \sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right)$$

- x_n controls the center of each bump

Bandwidth

If $K(u)$ is a kernel, then for any $h > 0$

$$K_h(u) \triangleq \frac{1}{h} K\left(\frac{u}{h}\right)$$ (stretching the kernel)

can be used as a kernel too (verify the two properties yourself)

So general KDE is determined by both the kernel K and the bandwidth h

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K_h(x - x_n) = \frac{1}{Nh} \sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right)$$

- x_n controls the center of each bump
- h controls the width/variance of the bumps

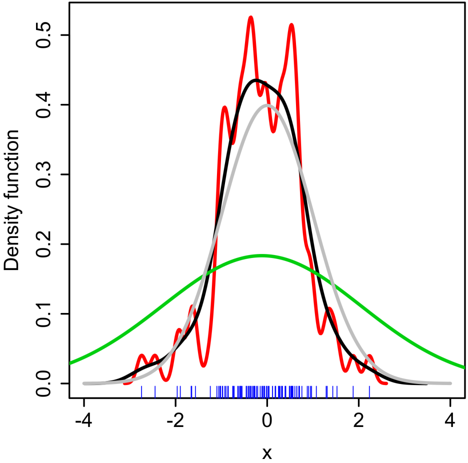
Effect of bandwidth

picture from Wikipedia

Larger h means larger variance and also smoother density

Gray curve is ground-truth

- Red: $h = 0.05$
- Black: $h = 0.337$
- Green: $h = 2$



Bandwidth selection

For selecting h

- there are theoretically-motivated approaches

Bandwidth selection

For selecting h

- there are theoretically-motivated approaches
- one can also do cross-validation based on downstream applications