Conditional Density Estimation via Least-Squares Density Ratio Estimation

Sugiyama, M., Takeuchi, I., Kanamori, T., Suzuki, T., Hachiya, H., & Okanohara, D. AISTATS 2010

(Arthur Gretton's notes)

January 24, 2013

What is the goal?

Given: samples

$$\left\{ z_i | z_i = (x_i, y_i) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \right\}_{i=1}^{n},$$

provide an estimate for

$$p(y|x) := \frac{p(x, y)}{p(x)} = r(x, y)$$

(i.e. given a test point x_{test}, get a function of y).

Assumption

Assume the estimate will take the form

$$\begin{aligned}\hat{r}_\alpha(x, y) &:= \alpha^\top \vec{\phi}(x, y) \\ &= \alpha^\top \left[\begin{array}{ccc} \phi_1(x, y) & \dots & \phi_b(x, y) \end{array} \right].\end{aligned}$$

Assumption

Assume the estimate will take the form

$$\hat{r}_\alpha(x, y) := \alpha^\top \vec{\phi}(x, y) = \alpha^\top \begin{bmatrix} \phi_1(x, y) & \dots & \phi_b(x, y) \end{bmatrix}.$$

In practice: use

$$\phi_\ell(x, y) := \exp\left(\frac{-\|x - u_\ell\|}{2\sigma^2}\right) \exp\left(\frac{-\|y - v_\ell\|}{2\sigma^2}\right) = k(x, u_l) k(y, v_l),$$

where (u_ℓ, v_ℓ) "randomly chosen from" $\left\{z_i | z_i = (x_i, y_i) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}\right\}_{i=1}^{n}$.
More generally: require $\phi_\ell \geq 0$.

Loss function

The loss function to optimize is:

$$J(\alpha) := \frac{1}{2} \int \int \left(\hat{r}_\alpha(x,y) - r(x,y) \right)^2 p(x) dx dy.$$

(note asymmetry in integral). Expand this out:

$$\begin{aligned} J(\alpha) = &\frac{1}{2} \int \int \hat{r}_\alpha^2(x,y) p(x) dx dy \\ &- \int \int \hat{r}_\alpha(x,y) \underbrace{r(x,y) p(x)}_{p(x,y)} dx dy \\ &+ C \end{aligned}$$

Loss function (continued)

The following is "semi-empirical": we substitite the expression for $\hat{r}_\alpha(x, y)$:

$$J(\alpha) := \frac{1}{2}\alpha^\top H\alpha - h^\top\alpha + C$$

where

$$H := \int \underbrace{\left[\int \vec{\phi}(x, y)\vec{\phi}^\top(x, y)dy\right]}_{\bar{\Phi}(x)} p(x)dx,$$

$$h := \int \vec{\phi}(x, y)p(x, y)dxdy.$$

Empirical loss

The empirical loss function is:

$$\hat{J} := \frac{1}{2}\alpha^\top \widehat{H}\alpha - \hat{h}^\top \alpha + \underbrace{\lambda\|\alpha\|^2}_{\text{regularizer}}$$

where

$$\widehat{H} = \frac{1}{n}\sum_{i=1}^{n} \bar{\Phi}(x_i) \qquad \hat{h} = \frac{1}{n}\sum_{i=1}^{n} \vec{\phi}(x_i, y_i)$$

The (q, r)th entry of $\bar{\Phi}(x_i)$ is

$$\bar{\Phi}_{q,r}(x_i) = k(x_i, u_q)k(x_i, u_r)\underbrace{\int k(y, v_q)k(y, v_r)dy}_{\propto \exp(-\sigma^{-1}\|v_q - v_r\|)}$$

Solution

The solution is simple:

$$\tilde{\alpha} = \left(\widehat{H} + \lambda I\right)^{-1} \hat{h}.$$

Solution

The solution is simple:

$$\tilde{\alpha} = \left(\widehat{H} + \lambda I\right)^{-1} \hat{h}.$$

But...need to enforce non-negativity (claim: similar result to a Q.P. with $\alpha \succeq 0$):

$$\hat{\alpha} := \max(0, \tilde{\alpha})$$

Also: need to renormalize at test point:

$$\hat{p}(y|x = x_{test}) = \frac{\hat{\alpha}^\top \vec{\phi}(\tilde{x}, y)}{\int \hat{\alpha}^\top \vec{\phi}(\tilde{x}, y) dy}.$$

Parameter tuning

Cross validate with negative log-likelihood:

$$\mathrm{NLL} := -\frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \log \hat{p}(\tilde{y}_i | \tilde{x}_i).$$

on validation set $\{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{\tilde{n}}$.

Parameter tuning

Cross validate with negative log-likelihood:

$$\mathrm{NLL} := -\frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \log \hat{p}(\tilde{y}_i | \tilde{x}_i).$$

on validation set $\{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{\tilde{n}}$.
Open problem: what happens when you cross-validate using the squared loss?

$$J(\alpha) := \frac{1}{2}\alpha^\top \widetilde{H}\alpha - \tilde{h}^\top \alpha,$$

where expectations for $\widetilde{H}$, $\tilde{h}$ taken over validation set.

Competing methods

- Vanilla ratio of Parzen windows
- Density from quantile regression (only for y in 1-D)
- Nearest neighbor:

$$\hat{p}(y|x) = \frac{1}{|\mathcal{I}_{x,\epsilon}|} \sum_{i \in \mathcal{I}_{x,\epsilon}} N\left(y; y_i, \sigma^2 I_{d_y}\right).$$

- Neural networks:

$$\hat{p}(y|x) = \sum_{\ell=1}^{t} \pi_\ell(x) N(y; \mu_\ell(x), \sigma_\ell^2(x)),$$

where weights, means, and variances learned by neural networks (cross validation over t "unbearably slow").

Does it work?

Dataset	(n, d_X)	LS-CDE	ϵ-KDE	MDN	KQR	RKDE
caution	(50,2)	**1.24 ± 0.29**	**1.25 ± 0.19**	**1.39 ± 0.18**	**1.73 ± 0.86**	17.11 ± 0.25
ftcollinssnow	(46,1)	**1.48 ± 0.01**	**1.53 ± 0.05**	**1.48 ± 0.03**	**2.11 ± 0.44**	46.06 ± 0.78
highway	(19,11)	**1.71 ± 0.41**	**2.24 ± 0.64**	7.41 ± 1.22	5.69 ± 1.69	15.30 ± 0.76
heights	(687,1)	**1.29 ± 0.00**	1.33 ± 0.01	**1.30 ± 0.01**	**1.29 ± 0.00**	54.79 ± 0.10
sniffer	(62,4)	**0.69 ± 0.16**	**0.96 ± 0.15**	**0.72 ± 0.09**	**0.68 ± 0.21**	26.80 ± 0.58
snowgeese	(22,2)	**0.95 ± 0.10**	1.35 ± 0.17	**2.49 ± 1.02**	**2.96 ± 1.13**	28.43 ± 1.02
ufc	(117,4)	**1.03 ± 0.01**	1.40 ± 0.02	**1.02 ± 0.06**	**1.02 ± 0.06**	11.10 ± 0.49
birthwt	(94,7)	**1.43 ± 0.01**	1.48 ± 0.01	**1.46 ± 0.01**	1.58 ± 0.05	15.95 ± 0.53
crabs	(100,6)	-0.07 ± 0.11	0.99 ± 0.09	**-0.70 ± 0.35**	**-1.03 ± 0.16**	12.60 ± 0.45
GAGurine	(157,1)	**0.45 ± 0.04**	0.92 ± 0.05	**0.57 ± 0.15**	**0.40 ± 0.08**	53.43 ± 0.27
geyser	(149,1)	**1.03 ± 0.00**	1.11 ± 0.02	1.23 ± 0.05	1.10 ± 0.02	53.49 ± 0.38
gilgais	(182,8)	0.73 ± 0.05	1.35 ± 0.03	**0.10 ± 0.04**	0.45 ± 0.15	10.44 ± 0.50
topo	(26,2)	**0.93 ± 0.02**	1.18 ± 0.09	2.11 ± 0.46	2.88 ± 0.85	10.80 ± 0.35
BostonHousing	(253,13)	0.82 ± 0.05	1.03 ± 0.05	**0.68 ± 0.06**	**0.48 ± 0.10**	17.81 ± 0.25
CobarOre	(19,2)	**1.58 ± 0.06**	**1.65 ± 0.09**	**1.63 ± 0.08**	6.33 ± 1.77	11.42 ± 0.51
engel	(117,1)	**0.69 ± 0.04**	1.27 ± 0.05	**0.71 ± 0.16**	N.A.	52.83 ± 0.16
mcycle	(66,1)	**0.83 ± 0.03**	1.25 ± 0.23	1.12 ± 0.10	**0.72 ± 0.06**	48.35 ± 0.79
BigMac2003	(34,9)	**1.32 ± 0.11**	**1.29 ± 0.14**	**2.64 ± 0.84**	**1.35 ± 0.26**	13.34 ± 0.52
UN3	(62,6)	**1.42 ± 0.12**	1.78 ± 0.14	**1.32 ± 0.08**	**1.22 ± 0.13**	11.43 ± 0.58
cpus	(104,7)	1.04 ± 0.07	1.01 ± 0.10	**-2.14 ± 0.13**	N.A.	15.16 ± 0.72
Time		1	0.004	267	0.755	0.089

Figure: Density ratio results

Does it work in theory?

Good question....