MLE and MAP

> Pattern Recognition Homeworks
>
> Student: thu-zxs

Solutions

1. **MLE for the uniform distribution**

   a.

   Assume $a > \max_{i=1,...,n}|x_i|$, then Maximum Likelihood Function $l(a)$:

$$
\begin{aligned}
l(a) &= \sum_{i=1}^{n} \log(x_i|a) \\
&= \sum_{i=1}^{n} \frac{1}{2a} \\
&= \frac{n}{2a}
\end{aligned}
$$

   Then $l(a)$ reaches maximum when $a = \max_{i=1,...,n}|x_i|$, **i.e.**

$$
\hat{a} = \max_{i=1,...,n}|x_i|
$$

   b.

   Obviously, if $x_{n+1} \in [-\hat{a}, \hat{a}]$, $p(x_{n+1}) = 0$, otherwise, $p(x_{n+1}) = \frac{1}{2\hat{a}}$

   c.

   By MLE, the paramater a is decided by the max absolute value of $\{x_i\}$, this estimation is biased from below deduction:

$$
\begin{aligned}
\hat{a} &= \max_{i=1,...,n}|x_i| \\
P(\hat{a} < x) &= P(\hat{x} < x)^n \\
P(\hat{x} < x) &= \begin{cases} 0, x \leq -a \\ \frac{x}{2a}, -a \leq x \leq a \\ 1, x \geq a \end{cases}
\end{aligned}
$$

   Then,

$$
\begin{aligned}
p(\hat{a}) = P(\hat{a} < x)' = nP(\hat{x} < x)^{n-1}p(x) \\
= \begin{cases} n\frac{x^{n-1}}{(2a)^n}, -a \leq x \leq a \\ 0, otherwise \end{cases}
\end{aligned}
$$

   So,

$$\mathbb{E}(\hat{a}) = \int_{-a}^{a} xn\frac{x^{n-1}}{(2a)^n}dx = \frac{n}{n+1}a \neq a$$

The estimation is biased.

Better approach is to use the first-order absolut Moment Estimation method：

$$\mathbb{E}(|x|) = \int_{-a}^{a} \frac{|x|}{2a}dx = \frac{a}{2}$$

Then, take $\hat{a} = 2\mathbb{E}|\mathrm{x}|$, which is an unbiased estimation.

2. Consider a training data of N i.i.d observations, $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ with corresponding N target values $\mathbf{T} = t_1, t_2, \ldots, t_N$.

   We want to fit these observations into some model

   $$t = y(x, \mathbf{w}) + \epsilon$$

   where $\mathbf{w}$ is the model parameters and ϵ is some error term.

   2.1

   maximize gaussian random variables probability $\prod p(t_n)$, **i.e.** make t_n close enough to $y(x_n, \mathbf{w})$.

   $$\begin{aligned} \max_w p(\mathbf{T}|\mathbf{X}, \mathbf{w}, \beta) &:= \max_w \sum_{n=1}^{N} \ln(\mathcal{N}(t_n|y(x_n, \mathbf{w}), \beta^{-1})) \\ &:= \min_w \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 \end{aligned}$$

   2.2

   $$\begin{aligned} \max_w p(w|X, T, \alpha, \beta) &:= \max_w p(\mathbf{T}|\mathbf{X}, \mathbf{w}, \beta)p(w|\alpha) \\ &:= \max_w \sum_{n=1}^{N} \ln(\mathcal{N}(t_n|y(x_n, \mathbf{w}), \beta^{-1})) + \ln(p(w|\alpha)) \\ &:= \min_w \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + |w|^2 \end{aligned}$$

Programming

3.

> Note:
>
> Use **n=10000** except for problem **c**
>
> Programming language: *Python*

For **Parzen** window:

$$P(x) = \begin{cases} \frac{1}{a}, -\frac{1}{2} \leq a \leq \frac{1}{2}a \\ 0, otherwise \end{cases}$$

Number of samples that lie in the cube around x with side length a:

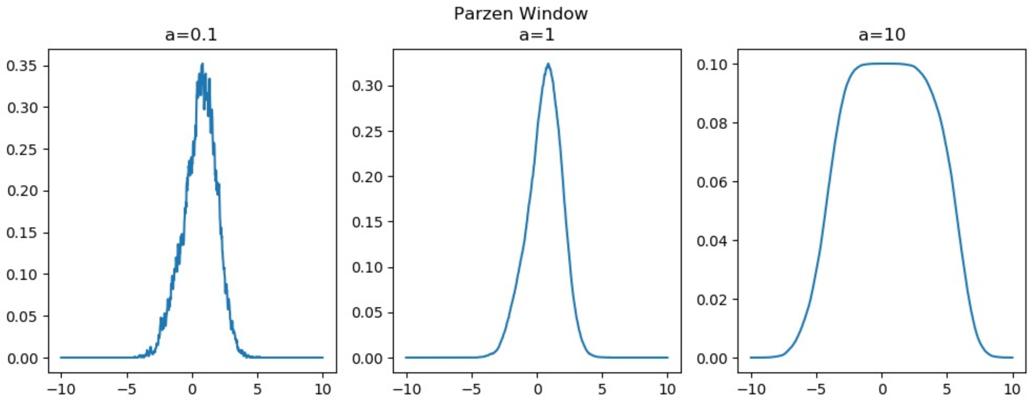
$$K = \sum_{n=1}^{N} a * P(a\frac{x - x_n}{a}) = \sum_{n=1}^{N} a * P(x - x_n)$$

Then,

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

a)

```
def draw_pnx_pdf_with_parzen_window(samples, a):
    """ 根据a值绘出Parzen窗估计的概率密度函数p_n(x)
    """
    x = np.linspace(LR, RR, num)
    N = samples.shape[0]
    tmp = x[:, np.newaxis] - samples
    px = np.sum(1.0/a*((tmp<=0.5*a) & (tmp>=-0.5*a)).astype(np.int), axis=1)/N
    # plt.plot(x,px)
    # plt.show()
    return x, px
```



b)

```
def gaussian(x, mu, sigma):
    """ 高斯pdf
    """
    y = np.exp(-(x - mu) ** 2 / (2 * sigma ** 2)) / (sigma * np.sqrt(2 *
np.pi))
```

```
    return y

def draw_px_pdf(mu1, sigma1, mu2, sigma2, alpha=0.2):
    """ 高斯混合pdf, 即px的pdf
    """
    x = np.linspace(LR, RR, num)
    px = alpha*gaussian(x, mu1, sigma1) + (1-alpha)*gaussian(x, mu2, sigma2)
    return x, px

def compute_epsilon(samples, window_type='Parzen'):
    """ b)问
    """
    if window_type == "Parzen":
        x, pnx = draw_pnx_pdf_with_parzen_window(samples, a=1)

    elif window_type == 'Gaussian':
        x, pnx = draw_pnx_pdf_with_gaussian_window(samples, a=1)

    _, px = draw_px_pdf(-1, 1, 1, 1)
    epsilon = np.sum((px[1:]-pnx[1:])**2*delta)

    return epsilon
```

c)

```
def compute_exp_and_var_for_epsilon(window_type='Parzen'):
    """ c)问
    """

    # Fix n
    samples = gen2mixtures(-1, 1, 1, 1, n_samples=10000, alpha=0.2)
    A = [0.02, 0.1, 1, 10, 50]
    res1 = []
    for a in A:
        res1.append(compute_epsilon(samples, window_type, a))
    res1 = np.asarray(res1)
    mean1 = np.mean(res1)
    var1 = np.var(res1)

    # Fix a
    N = [100, 500, 1000, 5000, 10000]
    a = 1
    res2 = []
    for n in N:
        samples = gen2mixtures(-1, 1, 1, 1, n_samples=n, alpha=0.2)
        res2.append(compute_epsilon(samples, window_type, a))
    res2 = np.asarray(res2)
```

```
    mean2 = np.mean(res2)
    var2 = np.var(res2)

    return mean1, var1, mean2, var2
```

fix $n = 10000$ and take $a = 0.02, 0.1, 1, 10, 50$, respectively:

$$\mathbb{E}_a(\epsilon(p_n)) = 0.0625$$
$$Var_a(\epsilon(p_n)) = 0.0061$$

fix $a = 1$ and take $n = 100, 500, 1000, 5000, 10000$ respectively:

$$\mathbb{E}_n(\epsilon(p_n)) = 0.0023$$
$$Var_n(\epsilon(p_n)) = 7.06 \times 10^{-6}$$

d)

In experiments, I computed $\epsilon(p_n)$ for $a = 0.02, 0.1, 1, 10, 50$ in each fixed $n = 5, 10, 100, 1000, 10000$ and chose the best a i.e. with the minimum $\epsilon(p_n)$:

n	**5**	**10**	**100**	**1000**	**10000**
optimal a	10	10	1	1	1
$\epsilon(p_n)$	0.121	0.116	0.0026	0.0015	0.0003

So from the experiments, when n is getting larger, we might choose the smaller a for optimum. But a median a is better for a more precise and smooth probabilistic model.

---

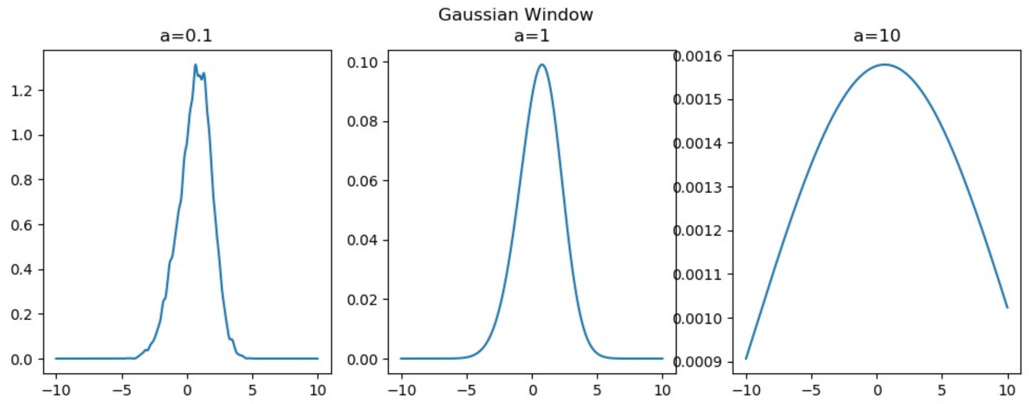
e)

For **Gaussian** window:

$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi a^2)}\exp\{-\frac{|x - x_n|^2}{2a^2}\}$$

a)

```python
def draw_pnx_pdf_with_gaussian_window(samples, a):
    """ 根据a值绘出Gaussian窗估计的概率密度函数p_n(x)
    """
    x = np.linspace(LR, RR, num)
    N = samples.shape[0]
    tmp = x[:, np.newaxis] - samples
    sigma = a**2
    px = np.sum(1.0/(2*np.pi*sigma)*np.exp(-(tmp)**2/(2*sigma)), axis=1)/N
    # plt.plot(x,px)
    # plt.show()
    return x, px
```



b)

See b) in **Parzen** window.

c)

code is shown in c) of **Parzen** window part

fix $n = 10000$ and take $a = 0.02, 0.1, 1, 10, 50$, respectively:

$$\mathbb{E}_a(\epsilon(p_n)) = 16.92$$
$$Var_a(\epsilon(p_n)) = 1060.81$$

fix $a = 1$ and take $n = 100, 500, 1000, 5000, 10000$ respectively:

$$\mathbb{E}_n(\epsilon(p_n)) = 0.09716$$
$$Var_n(\epsilon(p_n)) = 7.13 \times 10^{-7}$$

d)

In experiments, I computed $\epsilon(p_n)$ for $a = 0.02, 0.1, 1, 10, 50$ in each fixed $n = 5, 10, 100, 1000, 10000$ and chose the best a i.e. with the minimum $\epsilon(p_n)$:

n	**5**	**10**	**100**	**1000**	**10000**
optimal a	1	1	1	1	1
$\epsilon(p_n)$	0.155	0.085	0.099	0.099	0.097

From the experiments above, the best a is always a median one. Also a median a is better for a more precise and smooth probabilistic model.

h)

In my above experiments, the best (smooth and precise) model is the **Parzen Window** with following parameters set:

$$
\begin{align}
a &= 1 \\
N &= 10000
\end{align}
$$

$$
\begin{aligned}
a &= 1 \\
N &= 10000
\end{aligned}
$$