Homework 3

Chen Lu

June 4, 2018

1 Dimension Reduction, PCA

1.1 Minimum Error Formulation

A set of complete orthonormal basis

$$\{\mu_i\}, i = 1, ..., p \quad \mu_i^T \mu_j = \delta_{ij}$$

The best approximation is to minimize the error

$$J = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2 \tag{1.1}$$

$$x_n = \sum_i (x_n^T \mu_i) \mu_i \tag{1.2}$$

$$\tilde{x}_n = \sum_{i=1}^{d} z_{ni} \mu_i + \sum_{i=d+1}^{p} b_i \mu_i \tag{1.3}$$

Proof $z_{ni} = x_n^T \mu_i, i = 1, ..., d$ and $b_i = \bar{x}^T \mu_i, i = d+1, ..., p$
Proof:
We apply (1.2) and (1.3) into (1.1), and we get

$$J = \frac{1}{N}\sum_{n=1}^{N}\left\|\sum_{i=1}^{p}(x_n^T\mu_i)\mu_i - \sum_{i=1}^{d} z_{ni}\mu_i - \sum_{i=d+1}^{p} b_i\mu_i\right\|^2 \tag{1.4}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left[\sum_{i=1}^{d}((x_n^T\mu_i)\mu_i - z_{ni}\mu_i)^2 + \sum_{i=d+1}^{p}((x_n^T\mu_i)\mu_i - b_i\mu_i)^2\right] \tag{1.5}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d}((x_n^T\mu_i)\mu_i - z_{ni}\mu_i)^2 + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=d+1}^{p}((x_n^T\mu_i)\mu_i - b_i\mu_i)^2 \tag{1.6}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d}(x_n^T\mu_i - z_{ni})^2 + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=d+1}^{p}(x_n^T\mu_i - b_i)^2 \tag{1.7}$$

Calculate the gradient of (1.6)'s left term for z_{ni}, let it be 0. We can get

$$\frac{\partial J}{\partial z_{ni}} = \frac{2}{N}(x_n^T\mu_i - z_{ni})\cdot(-z_{ni}) = 0 \tag{1.8}$$

$$z_{ni} = x_n^T\mu_i, i = 1, ..., d \tag{1.9}$$

Calculate the gradient of (1.6)'s right term for b_i, let it be 0. We can get

$$\frac{\partial J}{\partial b_i} = \frac{2}{N}\sum_{n=1}^{N}(x_n^T\mu_i - b_i)\cdot(-b_i) = 0 \tag{1.10}$$

$$b_i = \frac{1}{N}\sum_{n=1}^{N} x_n^T\mu_i = \bar{x}^T\mu_i, i = d+1, ..., p \tag{1.11}$$

1.2 MNIST

1. Data: MNIST dataset from tensorflow.examples.tutorials.mnist.input_data

	dimension
imgs	(55000, 784)
label	(55000, 1)

2. Dimension reduction (orginally 784)

shape	centering	without centering
30%	(55000, 87)	(55000, 53)
60%	(55000, 17)	(55000, 6)
90%	(55000, 5)	(55000, 1)

3. Chosen digit: 1 and 8

4. Result: Fig. 4

2 Learning Theory

2.1 VC Dimension

If the points is placed as Fig.2.1, Fig.2.2

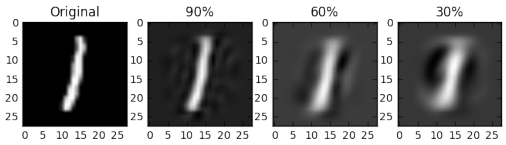

Figure 1.1: digit 1 with centering

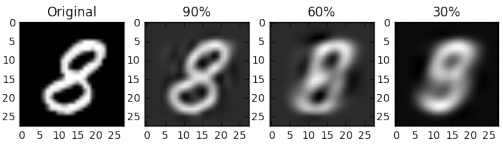

Figure 1.2: digit 8 with centering

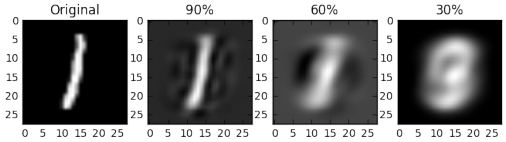

Figure 1.3: digit 1 without centering

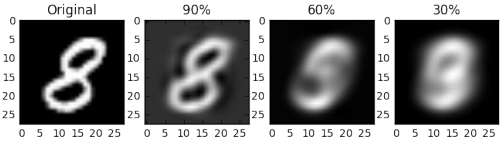

Figure 1.4: digit 8 without centering

1. VC Dimension ≥ 3

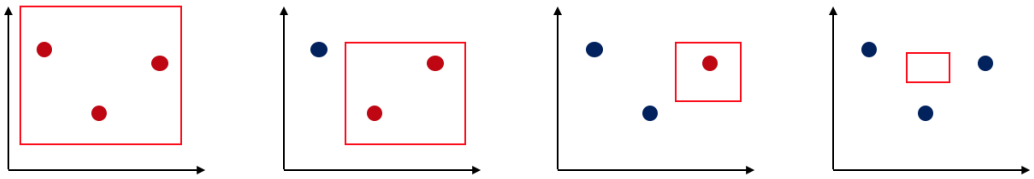

Figure 2.1: the graph of 3 points to be scattered

2. VC Dimension ≥ 4

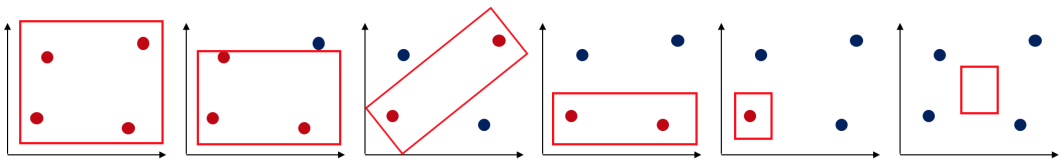

Figure 2.2: the graph of 4 points to be scattered

For any circumstance, we could divide nodes by axis-parallel rectangle. I show six different circumstances in Fig. 2.2 and we could extend to 24 easily. Thus, the VC dimension of axes-parallel rectangle is greater than 4.
Now, I will demonstrate next that the VC dimension is 4 exactly. For any arrangements of 5 points, we could find the points with maximum x coordinate, minimum x coordinate, maximum y coordinate, minimum y coordinate, as shown in fig.2.3

When these colored nodes own the same label classified as positive and other nodes's label is

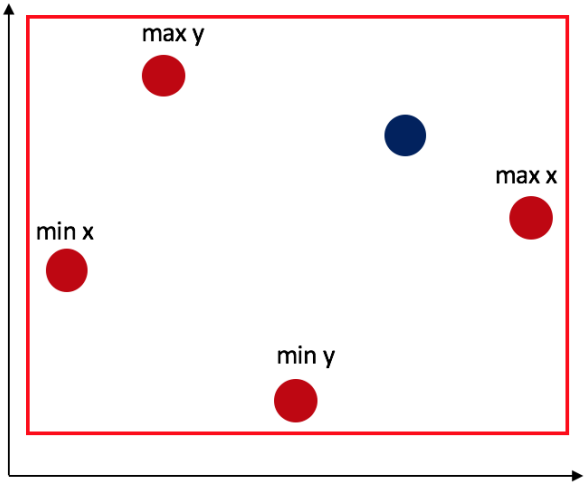

Figure 2.3: the graph of 5 points to be scattered

negative, we could not use an axes-parallel rectangle to contain them without adding other nodes.

2.2 Generalization Bound

I will use the VC Bound from the slides of Prof. Zhu, with probability at least $1-\delta$

$$\forall g \in G, R(g) \leq R_n(g) + 2\sqrt{2\frac{h\log\frac{2en}{h} + \log\frac{2}{\delta}}{n}} \tag{2.1}$$

In our circumstance, we have VC-dimension h = 2, sample number n = m and $R_n(g) = 0$. If we want an error of at least ϵ, then

$$\begin{aligned}\epsilon &= 2\sqrt{2\frac{h\log\frac{2en}{h} + \log\frac{2}{\delta}}{n}} \\ &= 2\sqrt{2\frac{2\log\frac{2em}{h} + \log\frac{2}{\delta}}{m}}\end{aligned}$$

And we could get that $\delta = 2e^2m^2\exp(-\frac{m\epsilon^2}{8})$. Then with probability at least $1-\delta$, we have $R(g) - R_n(g) \leq \epsilon$.
Finally, back to the question, the probability that a hypothesis $h \in H$ consistent with m instances $x_1, ..., x_m$ have an error of at least ϵ is $2e^2m^2\exp(-\frac{m\epsilon^2}{8})$

3 Reinforcement Learning

3.1 Value Iteration

$$
\begin{aligned}
Q &= R(s,a) + 0.9 \times (P(Non|s_t) \times (-1) + P(Light|s_t) \times (35) + P(Heavy|s_t) \times (50)) \\
0.72 &= 0 + 0.9 \times (0.95 \times (-1) + 0.05 \times (35) + 0.00 \times (50)) \\
45.695 &= 20 + 0.9 \times (0.2 \times (-1) + 0.75 \times (35) + 0.005 \times (50)) \\
137.71 &= 100 + 0.9 \times (0.1 \times (-1) + 0.2 \times (35) + 0.7 \times (50)) \\
52.71 &= 15 + 0.9 \times (0.1 \times (-1) + 0.2 \times (35) + 0.7 \times (50)) \\
120.68 &= 80 + 0.9 \times (0.05 \times (-1) + 0.15 \times (35) + 0.8 \times (50))
\end{aligned}
$$

3.2 Policy Improvement Theorem

$$
Q^{\pi}(s,a) = E_{\pi}\left\{r_{k+1} + \gamma V^{\pi}(s_{k+1}) \,|\, s_k = s, a_k = a\right\} \tag{3.1}
$$

The key criterion is whether this is greater than or less than $V^{\pi}(s)$. If it is greater–that is, if it is beeter to select a once in s and thereafter follow π than it would be to follow π all the time–then one would expect it to be better still to select a every time s is encountered, and that the new policy would in fact be a better one overall.

$$
V^{\pi_{k+1}} = Q^{\pi}(s,a) = Q^{\pi}(s,\pi_{k+1}) \geq V^{\pi_k} = Q^{\pi}(s,\pi_k) \tag{3.2}
$$

The policy π_{k+1} must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in S$.
If $Q^{\pi}(s,a) > V^{\pi_k}$, then the changed policy is indeed better than π.

$$
\begin{aligned}
V^{\pi_k} &\leq Q^{\pi}(s,\pi_{k+1}) \\
&= E_{\pi_{k+1}}\left\{r_{k+1} + \gamma V^{\pi_{k+1}} | s_k = s\right\} \\
&\leq E_{\pi_{k+1}}\left\{r_{k+1} + \gamma Q^{\pi}(s_{k+1},\pi_{k+1}) \,|\, s_k = s\right\} \\
&= E_{\pi_{k+1}}\left\{r_{k+1} + \gamma E_{\pi_{k+1}}\left\{r_{k+2} + \gamma V^{\pi_{k+2}}\right\} | s_k = s\right\} \\
&= E_{\pi_{k+1}}\left\{r_{k+1} + \gamma r_{k+2} + \gamma^2 V^{\pi_{k+2}} | s_k = s\right\} \\
&\leq E_{\pi_{k+1}}\left\{r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \gamma^3 V^{\pi_{k+3}} | s_k = s\right\} \\
&\ldots \\
&\leq E_{\pi_{k+1}}\left\{r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \gamma^3 r_{k+4} + \ldots | s_k = s\right\} \\
&= V^{\pi_{k+1}}
\end{aligned}
$$

3.3 TIANSHOU

Deep Q-learning Algorithm is a off-policy, value-based algorithm.
Taking ddpg.py as a reference, complete the deep Q-learning algorithm just running on CPU. When iteration is about 13000, i.e. running time is 24mins, the mean return converge to 86.60203 and mean undiscounted return is 200.0.
Here thanks for Li Zhihui's tianshou environment which helps me a lot :)