Machine Learning, Spring 2018
Homework 5

Due on 23:59 May 16, 2018
Lin Daquan
85610653

Understanding VC dimension

1. Suppose that $y_1 = +1, y_2 = +1, y_3 = -1, y_4 = +1$, have

$$\begin{aligned}
&\text{If } f(x_1) = y_1, f(x_2) = y_2, f(x_4) = y_4, \\
&\quad f(x_1) = \sin\alpha > 0 \\
&\quad f(x_2) = \sin 2\alpha = 2\sin\alpha\cos\alpha > 0, \Rightarrow \cos\alpha > 0 \\
&\quad f(x_3) = \sin 3\alpha = \sin\alpha\cos 2\alpha + \cos\alpha\sin 2\alpha \\
&\quad f(x_4) = \sin 4\alpha = 2\sin 2\alpha\cos 2\alpha > 0, \Rightarrow \cos 2\alpha > 0
\end{aligned} \tag{1}$$

we can deduce $f(x_3) > 0$, it contradicts with $y_3 < 0$

2. For $m > 0$, consider the set of points $(x_1, \ldots, x_m)$ with arbitrary labels $(y_1, \ldots, y_m) \in \{-1, +1\}^m$.
Suppose $x_i = 10^{-i}$, $\alpha = \pi(1 + \sum_{i=1}^{m} 10^i y_i')$, where $y_i' = \frac{1-y_i}{2}$.
For any $j \in [1, m]$, we have

$$\begin{aligned}
\sin\alpha x_j = \sin\alpha 10^{-j} &= \sin\pi(10^{-j} + \sum_{i=1}^{m} 10^{i-j} y_i') \\
&= \sin\pi(10^{-j} + \sum_{i=1}^{j-1} 10^{i-j} y_i' + y_j' + \sum_{i=1}^{m-j} 10^i y_i'), 0 \equiv \sum_{i=1}^{m-j} 10^i y_i' \bmod 2 \\
&= \sin\pi(10^{-j} + \sum_{i=1}^{j-1} 10^{i-j} y_i' + y_j') \\
&= \sin\pi(\sum_{i=1}^{j-1} 10^{-i} y_i' + 10^{-j} + y_j')
\end{aligned} \tag{2}$$

Since $y_i' \in \{0, 1\}$, upper and lower bound for $\pi(\sum_{i=1}^{j-1} 10^{-i} y_i' + 10^{-j} + y_j')$ as follows:

$$\pi y_j' < \pi(\sum_{i=1}^{j-1} 10^{-i} y_i' + 10^{-j} + y_j') \leq \pi(\sum_{i=1}^{j} 10^{-i} + y_j') < \pi(1 + y_j')$$

Thus, if $y_j = 1$, we have $y_j' = 0$ and $0 < \alpha x_j < \pi$, which implies $\text{sign}(\alpha x_j) = 1$. Similarly, for $y_j = 1$, we have $\text{sign}(\alpha x_j) = -1$.

Understanding Lasso (30 points)

1.

$$x^{k+1} = \arg\min_{\boldsymbol{x}} \{\frac{1}{2}\|Ax-b\|_2^2 + \rho \left\|Fx - z^k + \frac{1}{\rho}y^k\right\|_2^2\} \\ = (A^TA+\rho F^TF)^{-1}(A^Tb+\rho F^Tz^k - F^Ty^k) \tag{3}$$

$$z^{k+1} = \arg\min_{\boldsymbol{x}} \{\lambda \|z\|_1 + \frac{\rho}{2}\left\|Fx^{k+1-z+\frac{1}{\rho}y^K}\right\|_2^2\} \\ = S_{\frac{\lambda}{\rho}}(Fx^{k+1}+\frac{1}{\rho}y^k) \tag{4}$$

2. Please check code in `glasso.m`
   Result:
   iteration: 89, augment Lagrange multiplier rho: 12005.33 stopping criteron: deltaX0.000000, constraint 0.000000
   relative_error_x = $4.5290e-04$
   relative_error_z = $4.5290e-04$
   See more details in code.

3. **brain.png**
   Mean square error (MSE) : 0.0024867
   Peak signal-noise ratio (PSNR): 26.0438 dB

   lena.jpg
   Mean square error (MSE) : 0.0049722, 0.21814, 0.21814
   Peak signal-noise ratio (PSNR): 23.0346, 6.61264, 6.61264 dB

   phantom.png
   Mean square error (MSE) : 0.01004
   Peak signal-noise ratio (PSNR): 19.9825 dB

Figure 1: brain.png

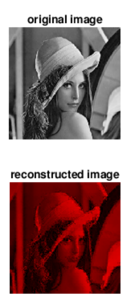

Figure 2: lena.jpg

Figure 3: phantom.png

Dual Formulation of the SVM (25 points)

1. Consider the soft-margin SVM:

$$\underset{w}{\text{minimize}}\ \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi^i, \text{subject to } y^i(w^Tx^i - b) \geq 1 - \xi^i \tag{5}$$

$$L(w,b,\xi,\alpha,\beta)=\frac{1}{2}\langle w,w\rangle+C\sum_{i=1}^{n}\xi^i-\sum_{i=1}^{n}\alpha^i(y^i(\langle w,x^i\rangle+b)-1+\xi^i)-\beta\xi^i$$

The KTT condition is:

$$\begin{aligned}\frac{\partial L}{\partial w}&=w-\sum_{i=0}^{n}y^i\alpha^i x^i=0\\ \frac{\partial L}{\partial b}&=-\sum_{i=1}^{n}\alpha^i y^i=0\\ \frac{\partial L}{\partial \xi}&=C-\alpha-\beta=0\end{aligned}\tag{6}$$

Therefore the dual fomulation is:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}}\ \frac{1}{2}\sum_{i,j}y^iy^j\alpha^i\alpha^j\langle x^i,x^j\rangle-\sum_{i=1}^{n}\alpha^i$$

2. - Pseudo code is shown in Fig.4
   - I split the $a3a$ into about 100 parts, and used $1,\dots,25$ parts to train the SVM gradually.Results is shown in Fig.5.
     Code is in folder: $Q3\_SMO$

Kernel function (25 points)

1. $\Phi:\mathbb{R}^2\to\mathbb{R}^3:(x_1,x_2)\mapsto(x_1^2,\sqrt{2}x_1x_2,x_2^2)$
   $\Phi(x_1,x_2)\cdot\Phi(x_1',x_2')=(x_1^2,\sqrt{2}x_1x_2,x_2^2)\cdot({x_1'}^2,\sqrt{2}x_1'x_2',{x_2'}^2)=x_1^2{x_1'}^2+2x_1x_1'x_2x_2'+x_2^2{x_2'}^2$
   Kernel function: $K(\boldsymbol{x},\boldsymbol{x'})=(\boldsymbol{x},\boldsymbol{x'})^2=((x_1,x_2)\cdot(x_1',x_2'))^2=(x_1x_1'+x_2x_2')^2=x_1^2{x_1'}^2+2x_1x_1'x_2x_2'+x_2^2{x_2'}^2$

2. (a) The SVM problem is defined as below:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}}\ \frac{1}{2}\sum_{i,j}y_iy_j\alpha_i\alpha_jK(\boldsymbol{x_i},\boldsymbol{x_j})-\sum_{i=1}^{n}\alpha_i$$

$$\text{subject to: }\boldsymbol{y^T\alpha}=0,0\le\boldsymbol{\alpha}\le C$$

$$\Rightarrow\boldsymbol{\alpha^*}$$

   Index s: $0<\alpha_s^*<C$

$$b^*=y_s-\sum_{\alpha_n^*>0}\alpha_n^*y_nK(\boldsymbol{x_n},\boldsymbol{x_s})$$

   The final hypothesis is:

$$g(\boldsymbol{x})=\text{sign}(\sum_{\alpha_n^*>0}\alpha_n^*y_nK(\boldsymbol{x_n},\boldsymbol{x})+b^*)$$

   The final hyperplane is :

$$x_1^2+x_2^2=\frac{5}{2}$$

   Separating hyperplane in Fig.6 Code in $Q4SVM\_polyKernel.py$.

   (b) If utilize the map above:$\Phi:\mathbb{R}^2\to\mathbb{R}^3$, project the train data into $\mathbb{R}^3$, along with a SVM.
   Then the hyperplane shown in Fig.7.
   The hyperplane function is:

$$z(x,y)=5-x$$

   Code in $Q4\_3D.py$.

Algorithm: Simplified SMO

Input:

C: regularization parameter
tol: numerical tolerance
max_passes: max # of times to iterate over α's without changing
$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$: training data

Output:

$\alpha \in \mathbb{R}^m$: Lagrange multipliers for solution
$b \in \mathbb{R}$: threshold for solution

- Initialize $\alpha_i = 0, \forall i, \quad b = 0$.
- Initialize $passes = 0$.
- **while** $(passes < max\_passes)$
  - $num\_changed\_alphas = 0$.
  - **for** $i = 1, \ldots m$,
    - Calculate $E_i = f(x^{(i)}) - y^{(i)}$ using (2).
    - **if** $((y^{(i)}E_i < -tol \;\; \&\& \;\; \alpha_i < C) \; || \; (y^{(i)}E_i > tol \;\; \&\& \;\; \alpha_i > 0))$
      - Select $j \neq i$ randomly.
      - Calculate $E_j = f(x^{(j)}) - y^{(j)}$ using (2).
      - Save old α's: $\alpha_i^{(\text{old})} = \alpha_i, \; \alpha_j^{(\text{old})} = \alpha_j$.
      - Compute L and H by (10) or (11).
      - **if** $(L == H)$
        continue to next i.
      - Compute η by (14).
      - **if** $(\eta >= 0)$
        continue to next i.
      - Compute and clip new value for α_j using (12) and (15).
      - **if** $(|\alpha_j - \alpha_j^{(\text{old})}| < 10^{-5})$
        continue to next i.
      - Determine value for α_i using (16).
      - Compute b_1 and b_2 using (17) and (18) respectively.
      - Compute b by (19).
      - $num\_changed\_alphas := num\_changed\_alphas + 1$.
    - **end if**
  - **end for**
  - **if** $(num\_changed\_alphas == 0)$
    $passes := passes + 1$
  - **else**
    $passes := 0$
- **end while**

Figure 4: SMO Algorithm

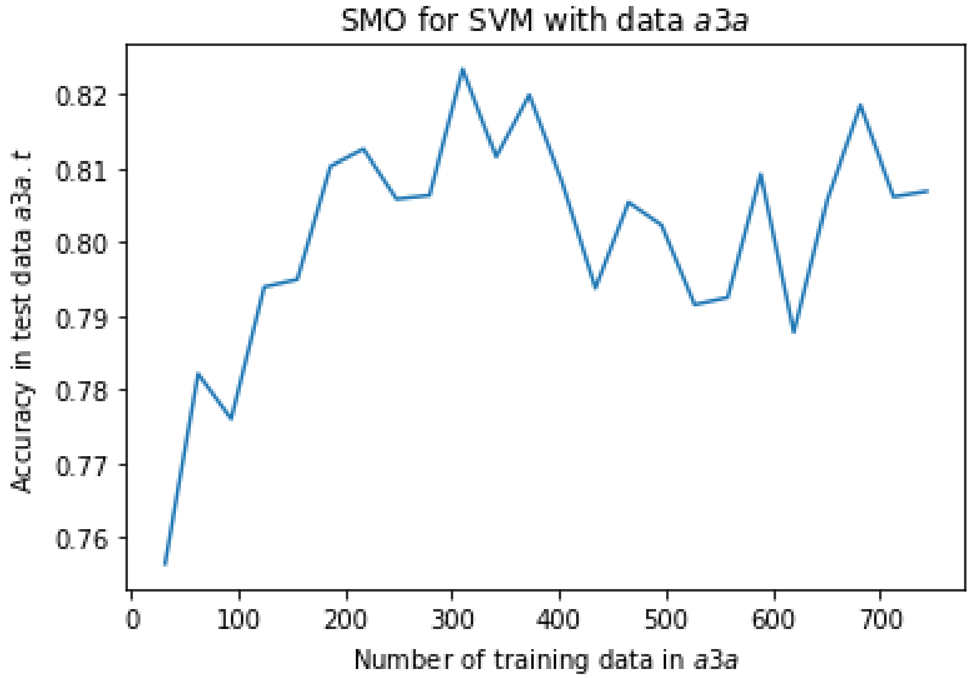

Figure 5: Accuracy with test data a3a.t.

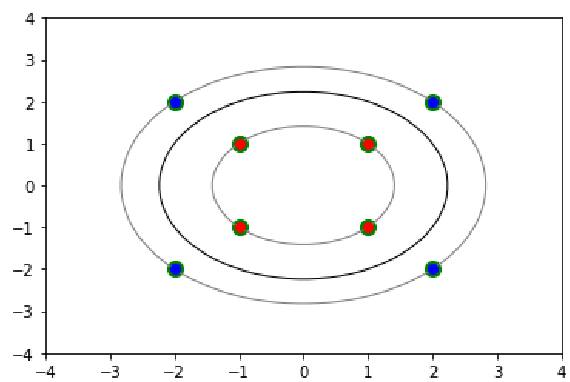

Figure 6: SVM classifier with polynomial(degree 2).

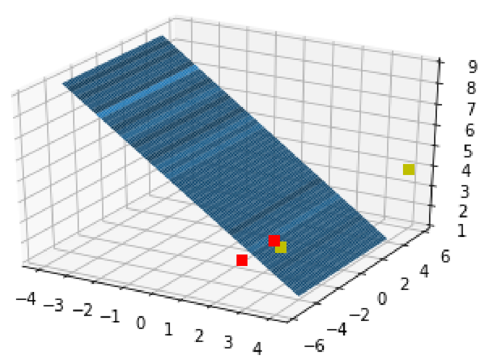

Figure 7: SVM classifier in 3D).