1.

EECS 545 Homework 4 Yuan Yin

1) Kernels

a) $K(\vec{u},\vec{v}) = (<u,v>+1)^3 = (<u,v>)^3 + 3(<u,v>)^2 + 3<u,v> + 1.$

$$= \sum_{(j_1\cdots j_d):\sum j_i=3} \binom{3}{j_1\cdots j_d} (u^{(1)})^{j_1}\cdots(u^{(d)})^{j_d}(v^{(1)})^{j_1}\cdots(v^{(d)})^{j_d}$$

$$+ 3\cdot\sum_{(k_1\cdots k_d):\sum k_i=2} \binom{2}{k_1\cdots k_d} (u^{(1)})^{k_1}\cdots(u^{(d)})^{k_d}(v^{(1)})^{k_1}\cdots(v^{(d)})^{k_d}$$

$$+ 3\sum_{i=1}^{d} u^{(i)}v^{(i)} + 1.$$

$$= <\Phi(u), \Phi(v)>.$$

Thus, $\Phi(u) = [\cdots, \sqrt{\binom{3}{j_1\cdots j_d}}(u^{(1)})^{j_1}\cdots(u^{(d)})^{j_d}, \cdots,$

$\cdots, \sqrt{\binom{2}{k_1\cdots k_d}}(u^{(1)})^{k_1}\cdots(u^{(d)})^{k_d}, \cdots,$

$\cdots, \sqrt{3}(u^{(1)}), \cdots, \sqrt{3}(u^{(d)}), 1]$

b) (i) $K(\vec{x},\vec{z}) = K_1(\vec{x},\vec{z}) + K_2(\vec{x},\vec{z})$

$= K_1(\vec{z},\vec{x}) + K_2(\vec{z},\vec{x}) = K(\vec{z},\vec{x}).$

$\Rightarrow K$ is symmetric

Suppose $A = B + C$, where $B = \begin{bmatrix} K_1(x_1,x_1) & \cdots & K_1(x_1,x_n) \\ \vdots & & \\ K_1(x_n,x_1) & \cdots & K_1(x_n,x_n)\end{bmatrix}.$

B is PSD matrix for K_1.

and C is PSD matrix for K_2. $C = \begin{bmatrix} K_2(x_1,x_1) & \cdots & K_2(x_1,x_n) \\ \vdots & & \\ K_2(x_n,x_1) & \cdots & K_2(x_n,x_n)\end{bmatrix}$

Thus, for $\forall \vec{z} \in R^n$.

$z^TAz = z^TBz + z^TCz \geq 0. \Rightarrow A$ is PSD matrix

Thus $K(\vec{x},\vec{z})$ is a symmetric positive-definite kernel.

由 扫描全能王 扫描创建

(ii) $k(\vec{x},\vec{z})$ may not be a positive-definite kernel.

let $k_2(\vec{x},\vec{z}) = 2k_1(\vec{x},\vec{z})$

Then $k(\vec{x},\vec{z}) = -k_1(\vec{x},\vec{z})$.

Let $A = \begin{bmatrix} k_1(x_1,x_1) & \cdots & k_1(x_1,x_n) \\ \vdots & & \vdots \\ k_1(x_n,x_1) & \cdots & k_1(x_n,x_n) \end{bmatrix}$ is the PSD matrix for k_1 ∴

Then $-A$ can't be PSD and thus $k(\vec{x},\vec{z})$ isn't a positive-definite kernel.

(iii) $k(\vec{x},\vec{z}) = a k_1(\vec{x},\vec{z}) = a k_1(\vec{z},\vec{x}) = k(\vec{z},\vec{x})$.

$\Rightarrow k(\vec{x},\vec{z})$ is symmetric.

Let $B = \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix} = a \cdot \begin{bmatrix} k_1(x_1,x_1) & \cdots & k_1(x_1,x_n) \\ \vdots & & \vdots \\ k_1(x_1,x_n) & \cdots & k_1(x_n,x_n) \end{bmatrix}$

$=: a \cdot A$.

Thus A is the PSD matrix for k_1

$\forall \vec{z} \in \mathbb{R}^n$. $z^T A z = a \cdot (z^T B z) \geq 0$. $(\because a \in \mathbb{R}^+)$

$\Rightarrow A$ is PSD matrix.

$\Rightarrow k$ is SPD kernel.

(iv). $k(\vec{x},\vec{z}) = k_1(\vec{x},\vec{z}) k_2(\vec{x},\vec{z}) = k_1(\vec{z},\vec{x}) \cdot k_2(\vec{z},\vec{x}) = k(\vec{z},\vec{x})$.

$\Rightarrow k$ is symmetric

Suppose $k_1(\vec{x},\vec{z}) = \Phi_1(x)^T \Phi_1(z)$ (since SPD kernel $\Leftrightarrow$ inner product kernel.)

$k_2(\vec{x},\vec{z}) = \Phi_2(x)^T \Phi_2(\vec{z})$

Where $\Phi_1(x) \in \mathbb{R}^n$ $\Phi_2(x) \in \mathbb{R}^m$, $\vec{x},\vec{z} \in \mathbb{R}^d$.

$\Rightarrow k(\vec{x},\vec{z}) = \langle \Phi_1(x), \Phi_1(z) \rangle * \langle \Phi_2(x), \Phi_2(z) \rangle$

$= \left(\sum_{i=1}^{n} \Phi_1^{(i)}(x) \Phi_1^{(i)}(z) \right) * \left(\sum_{j=1}^{m} \Phi_2^{(j)}(x) \Phi_2^{(j)}(z) \right)$

由 扫描全能王 扫描创建

$$k(\vec{x},\vec{z}) = \sum_{i=1}^{n}\sum_{j=1}^{m} \left(\Phi_1^{(i)}(x)\, \Phi_2^{(j)}(x)\right)\cdot\left(\Phi_1^{(i)}(z)\cdot\Phi_2^{(j)}(z)\right)$$

$\Rightarrow$ $k(\vec{x},\vec{z})$ is a Inner Product kernel

Thus $k(\vec{x},\vec{z})$ is a SPD kernel.

(v) $k(\vec{x},\vec{z}) = f(\vec{x})f(\vec{z}) = f(\vec{z})f(\vec{x}) = k(\vec{z},\vec{x})$

$\Rightarrow$ k is symmetric.

$$\mathbb{K} = \begin{bmatrix} k(\vec{x}_1,\vec{x}_1) & \cdots & k(\vec{x}_1,\vec{x}_n) \\ \vdots & & \vdots \\ k(\vec{x}_n,\vec{x}_1) & \cdots & k(\vec{x}_n,\vec{x}_n) \end{bmatrix}_{n\times n} = [f(x_1),\cdots,f(x_n)]^T\cdot[f(x_1),\cdots,f(x_n)]$$

$\Rightarrow$ $\forall \vec{z}\in\mathbb{R}^n$. $z^T\mathbb{K}z = z^T[f(x_1),\cdots,f(x_n)]^T[f(x_1),\cdots,f(x_n)]z$

$= \|z\cdot[f(x_1),\cdots,f(x_n)]\|^2 \geq 0.$

$\Rightarrow$ $\mathbb{K}$ is PSD matrix

Thus k is symmetric positive-definite kernel.

(vi). $k(\vec{x},\vec{z}) = P(k_1(\vec{x},\vec{z})) = P(k_1(\vec{z},\vec{x})) = k(\vec{z},\vec{x})$.

$\Rightarrow$ k is symmetric.

According conclusion of (iv). let $k_2(x,z) = k_1(x,z)$.

Then $k(x,z) = k_1^2(x,z)$ is SPD kernel.

By induction, $k(x,z) = k_1^n(x,z)$ is still SPD kernel.

According conclusion of (iii). $\forall a_n\in\mathbb{R}^+$.

$k(x,z) = a_n\cdot k_1^n(x,z)$ is still SPD kernel

According conclusion of (i). $\forall a_n, a_m\in\mathbb{R}^+$

$k(x,z) = a_n k_1^n(x,z) + a_m k_1^m(x,z)$ is still SPD kernel.

Thus, $k(\vec{x},\vec{z}) = P(k_1(\vec{x},\vec{z})) = \sum_{i=1}^{p} a_i k_1^i(x,z)$ $(a_i\in\mathbb{R}^+)$ is SPD kernel.

由 扫描全能王 扫描创建

(c). k is IP kernel $\Rightarrow$ k is SPD kernel

$$k(\vec{u},\vec{v}) = \langle \Phi(\vec{u}), \Phi(\vec{v}) \rangle = \langle \Phi(\vec{v}), \Phi(\vec{u}) \rangle = k(\vec{v},\vec{u})$$

$\Rightarrow$ k is symmetric.

Suppose $A = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \vdots \\ k(x_n,x_1) & & k(x_n,x_n) \end{pmatrix} = \begin{pmatrix} \Phi(x_1)^T\Phi(x_1) & \cdots & \Phi(x_1)^T\Phi(x_n) \\ \vdots & & \\ \Phi(x_n)^T\Phi(x_1) & \cdots & \Phi(x_n)^T\Phi(x_n) \end{pmatrix}$

$\forall \vec{z} \in \mathbb{R}^n$, $z^TAz = \sum_{i,j=1}^{n} z_i z_j \langle \Phi(x_i), \Phi(x_j) \rangle$

$$= \langle \sum_{i=1}^{n} z_i \Phi(x_i), \sum_{j=1}^{n} z_j \Phi(x_j) \rangle$$

$$= \| \sum_{i=1}^{n} z_i \Phi(x_i) \|^2 \geq 0.$$

$\Rightarrow$ A is PSD matrix.

Thus, k is SPD kernel.

2). kernel Ridge Regression.

(a). obj. fun $= \min_{\vec{w},b} \frac{1}{n} \sum_{i=1}^{n} (y_i - \vec{w}^T\vec{x_i} - b)^2 + \lambda \|\vec{w}\|^2$

solution: $\hat{w} = (\tilde{X}^T\tilde{X} + n\lambda I)^{-1} \tilde{X}^T \tilde{y}$

$\hat{b} = \bar{y} - \hat{w}^T \vec{\bar{x}}$

where $\tilde{y} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_n \end{bmatrix}$, $\tilde{X} = \begin{bmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_n^T \end{bmatrix}$

$\tilde{x_i} = \vec{x_i} - \bar{x} \in \mathbb{R}^d$.

$\tilde{y_i} = y_i - \bar{y} \in \mathbb{R}$.

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} \vec{x_i}$

$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$

Thus the obj. fun $= \| \tilde{y} - \tilde{X}w \|^2 + n\lambda \|w\|^2$

According to the lemma in notes.

$$(\tilde{X}^T\tilde{X} + n\lambda I)^{-1} = \frac{1}{n\lambda} [I - \tilde{X}^T (\tfrac{1}{n\lambda} I + \tilde{X}\tilde{X}^T)^{-1} \tilde{X}]$$

$$\Rightarrow \hat{w} = (\tilde{X}^T\tilde{X} + n\lambda I)^{-1} \tilde{X}^T \tilde{y} = \frac{1}{n\lambda} [\tilde{X}^T - \tilde{X}^T (n\lambda I + \tilde{X}\tilde{X}^T)^{-1} \tilde{X}\tilde{X}^T] \tilde{y}$$

由 扫描全能王 扫描创建

3.

$$\hat{w} = \frac{1}{n\lambda}\left[\tilde{X}^T - \tilde{X}^T(\tilde{G} + n\lambda I)^{-1}\tilde{G}\right]\tilde{y}$$

where $\tilde{G} = \tilde{X}\tilde{X}^T$.

$$\hat{b} = \bar{y} - \hat{w}^T\bar{x} = \bar{y} - \frac{1}{n\lambda}\tilde{y}^T[\tilde{X} - \tilde{G}(\tilde{G}+n\lambda I)^{-1}\tilde{X}]\bar{x}$$

$$= \bar{y} - \tilde{y}^T(\tilde{G}+n\lambda I)^{-1}\tilde{X}\bar{x}$$

$$= \bar{y} - \tilde{y}^T(\tilde{G}+n\lambda I)^{-1}\tilde{g}(\bar{x})$$

where $\tilde{G} = \begin{bmatrix} \langle \tilde{x}_1, \tilde{x}_1\rangle & \cdots & \langle \tilde{x}_1, \tilde{x}_n\rangle \\ \vdots & & \\ \langle \tilde{x}_n, \tilde{x}_1\rangle & \cdots & \langle \tilde{x}_n, \tilde{x}_n\rangle \end{bmatrix}$ $\quad \tilde{g}(\bar{x}) = \begin{bmatrix} \langle \tilde{x}_1, \bar{x}\rangle \\ \vdots \\ \langle \tilde{x}_n, \bar{x}\rangle \end{bmatrix}$

substitute $\langle \tilde{x}_i, \tilde{x}_j\rangle$ with $\langle \tilde{\Phi}(x_i), \tilde{\Phi}(x_j)\rangle$.

To kernelize, observe: $\langle \tilde{x}_i, \tilde{x}_j\rangle = \langle x_i - \bar{x}, x_j - \bar{x}\rangle$

$$= \langle x_i, x_j\rangle - \frac{1}{n}\sum_{r=1}^{n}\langle x_i, x_r\rangle - \frac{1}{n}\sum_{s=1}^{n}\langle x_s, x_j\rangle + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n}\langle x_r, x_s\rangle$$

and $\langle \tilde{x}_i, \bar{x}\rangle = \langle x_i - \bar{x}, \bar{x}\rangle$

$$= \frac{1}{n}\sum_{r=1}^{n}\langle x_i, x_r\rangle - \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n}\langle x_r, x_s\rangle$$

Now substitute $\langle x, x'\rangle$ with $k(x, x')$, Define the mean-centered feature map

$$\tilde{\Phi}(x) := \Phi(x) - \frac{1}{n}\sum_{l=1}^{n}\Phi(x_l)$$

$\Rightarrow \hat{b} = \bar{y} - \tilde{y}^T(\tilde{K} + n\lambda I)^{-1}\tilde{k}(\bar{x})$ where $\tilde{K} = [\langle \tilde{\Phi}(x_i), \tilde{\Phi}(x_j)\rangle]_{1\le i,j\le n}$

$$\tilde{k}(\bar{x}) = \begin{bmatrix} \langle \tilde{\Phi}(x_1), \frac{1}{n}\sum_{l=1}^{n}\Phi(x_l)\rangle \\ \vdots \\ \langle \tilde{\Phi}(x_n), \frac{1}{n}\sum_{l=1}^{n}\Phi(x_l)\rangle \end{bmatrix}$$

由 扫描全能王 扫描创建

(b). $\tilde{\mathbb{K}} = \mathbb{K} - \mathbb{K}O - O\mathbb{K} + O\mathbb{K}O$

Determine that: $\tilde{\mathbb{K}}'_{n\times m} = \mathbb{K}' - \mathbb{K}O_1 - O_2\mathbb{K}' + O_2\mathbb{K}O_1$

where : O_1 is a $n\times m$ matrix where all entries equal to $\frac{1}{n}$

O_2 is a $n\times n$ matrix where all entries equal to $\frac{1}{n}$

To check this: $\tilde{\mathbb{K}}'_{ij} = \langle \tilde{\Phi}(x_i), \tilde{\Phi}(x_j') \rangle = (\Phi(x_i) - \frac{1}{n}\sum_{p=1}^{n} \Phi(x_p)) *$

$(\Phi(x_j') - \frac{1}{n}\sum_{q=1}^{n} \Phi(x_q))$.

$$= \Phi(x_i)\cdot\Phi(x_j') - \frac{1}{n}\Phi(x_i)\cdot\sum_{q=1}^{n}\Phi(x_q)$$

$$- \frac{1}{n}\Phi(x_j')\cdot\sum_{p=1}^{n}\Phi(x_p)$$

$$+ \frac{1}{n^2}\sum_{p,q=1}^{n}\Phi(x_p)\Phi(x_q)$$

$$= \mathbb{K}'_{ij} - (\mathbb{K}O_1)_{ij} - (O_2\mathbb{K}')_{ij} + (O_2\mathbb{K}O_1)_{ij} \quad \checkmark$$

prediction formula:

$\hat{f}(x_i') = \hat{w}^T x_i' + \hat{b} = \bar{y} + \tilde{y}^T(\tilde{\mathbb{K}} + n\lambda I)^{-1}\tilde{\mathbb{K}}(x_i')$ where $\tilde{\mathbb{K}}(x_i') = \begin{bmatrix} \langle\tilde{\Phi}(x_1), \tilde{\Phi}(x_i')\rangle \\ \vdots \\ \langle\tilde{\Phi}(x_n), \tilde{\Phi}(x_i')\rangle \end{bmatrix}$

$\cancel{\frac{1}{n\lambda}\tilde{y}^T[\tilde{X} - \tilde{\mathbb{K}}(\tilde{\mathbb{K}}+n\lambda I)^{-1}\tilde{X}]x_i'}$

$\hat{f}(X') = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}_{m\times 1} * \bar{y} + \tilde{\mathbb{K}}'^T(\tilde{\mathbb{K}} + n\lambda I)^{-1}\tilde{y}$

(where X' is $m\times d$ test data matrix)

3). Support Vector Regression

a. The obj. fun: $\frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-)$

let $t_i = w^T x_i - b$. Observe that if $\xi_i^+ > 0$, $y_i - t_i = \epsilon + \xi_i^+$

That is, the inequality constraint is an equality because increasing ξ_i^+ beyond equality unnecessarily increases the obj. fun. Since this equality can only hold if $y_i - t_i \geq \varepsilon$, $\xi_i^+ = \max\{0, y_i - t_i - \varepsilon\}$

Similary we may deduce that $\xi_i^- = \max\{0, t_i - y_i - \varepsilon\}$

由 扫描全能王 扫描创建

Note that if $\xi_i^+ > 0$ then $\xi_i^- = 0$, and vice versa. Thus,

$$\xi_i^- + \xi_i^+ = \max\{0, |y_i - t_i| - \varepsilon\},$$

and the optimization problem reduce to

$$\min_{\vec{w}, b} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max\{0, |y_i - t_i| - \varepsilon\}$$

or setting $\lambda = \frac{1}{2C}$

$$\min_{\vec{w}, b} \frac{1}{n}\sum_{i=1}^{n} l_\varepsilon(y_i, w^T x_i + b) + \lambda \|w\|^2$$

b. The optimization problem is:

$$\min_{\vec{w}, b, \vec{\xi}^+, \vec{\xi}^-} \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-)$$

$$\text{s.t. } y_i - w^T x_i - b \le \varepsilon + \xi_i^+ \quad \forall i$$
$$w^T x_i + b - y_i \le \varepsilon + \xi_i^- \quad \forall i$$
$$\xi_i^+ \geq 0 \quad \forall i$$
$$\xi_i^- \geq 0 \quad \forall i$$

Lagrangian: $L(\vec{w}, b, \vec{\xi}^+, \vec{\xi}^-, \alpha, \beta, \lambda, \gamma) = \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-)$

$$+ \sum_{i=1}^{n} \alpha_i (y_i - w^T x_i - b - \varepsilon - \xi_i^+)$$
$$+ \sum_{i=1}^{n} \beta_i (w^T x_i + b - y_i - \varepsilon - \xi_i^-)$$
$$- \sum_{i=1}^{n} \lambda_i \xi_i^+ - \sum_{i=1}^{n} \gamma_i \xi_i^-$$

Dual Function: $L_D(\alpha, \beta, \lambda, \gamma) = \min_{\vec{w}, b, \vec{\xi}^+, \vec{\xi}^-} L(\vec{w}, b, \vec{\xi}^+, \vec{\xi}^-, \alpha, \beta, \lambda, \gamma)$

orgional problem is quadratic w.r.t. $\vec{w}$, and linear w.r.t. $\vec{\xi}^+, \vec{\xi}^-$, it's convex, we can get optimization of Dual Function above by:

$$\frac{\partial L}{\partial w} = \vec{w} - \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \beta_i x_i \stackrel{!}{=} 0 \Rightarrow w^* = \sum_{i=1}^{n} \alpha_i^* x_i - \sum_{i=1}^{n} \beta_i^* x_i$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \beta_i \stackrel{!}{=} 0 \Rightarrow \sum_{i=1}^{n} \alpha_i^* = \sum_{i=1}^{n} \beta_i^*$$

由 扫描全能王 扫描创建

$$\frac{\partial L}{\partial \xi_i^+} = \frac{C}{n} - \alpha_i - \lambda_i \overset{!}{=} 0 \Rightarrow \frac{C}{n} = \lambda_i^* + \alpha_i^*$$

$$\frac{\partial L}{\partial \xi_i^-} = \frac{C}{n} - \beta_i - \gamma_i \overset{!}{=} 0 \Rightarrow \frac{C}{n} = \beta_i^* + \gamma_i^*$$

$$\Rightarrow L_D(\alpha, \beta, \lambda, \gamma) = \frac{1}{2}\|w^*\|_2^2 + \frac{C}{n}\sum_{i=1}^n (\xi_i^+ + \xi_i^-) + \sum_{i=1}^n \alpha_i^* (y_i - w^{*T}x_i - b - \varepsilon - \xi_i^+)$$
$$+ \sum_{i=1}^n \beta_i^* (w^{*T}x_i + b - y_i - \varepsilon - \xi_i^-) - \sum_{i=1}^n \lambda_i^* \xi_i^+ - \sum_{i=1}^n \gamma_i^* \xi_i^-$$

[In order to satisfy KKT condition, we still need:

$$\alpha_i^*(y_i - w^{*T}x_i^* - b - \varepsilon - \xi_i^+) = 0 \quad \forall i$$
$$\beta_i^*(w^{*T}x_i^* + b - y_i - \varepsilon - \xi_i^-) = 0 \quad \forall i$$

where x_i^* is primal optimal]

$$L_D(\alpha, \beta, \lambda, \gamma) = \frac{1}{2}\left\|\sum_{i=1}^n (\alpha_i^* - \beta_i^*) x_i\right\|^2 + \frac{C}{n}\sum_{i=1}^n \cancel{(\xi_i^+ + \xi_i^-)}$$
$$+ \sum_{i=1}^n \alpha_i^*(y_i - w^{*T}x_i \cancel{- b} - \varepsilon) - \sum_{i=1}^n \cancel{\alpha_i^* \xi_i^+} - \sum_{i=1}^n \cancel{\lambda_i^* \xi_i^+}$$
$$+ \sum_{i=1}^n \beta_i^*(w^{*T}x_i \cancel{+ b} - y_i - \varepsilon) - \sum_{i=1}^n \cancel{\beta_i^* \xi_i^-} - \sum_{i=1}^n \cancel{\gamma_i^* \xi_i^-}$$
$$= \frac{1}{2}\sum_{i,j=1}^n (\alpha_i^* - \beta_i^*)(\alpha_j^* - \beta_j^*) x_i^T x_j + \sum_{i=1}^n \alpha_i^*(y_i - w^{*T}x_i - \varepsilon)$$
$$+ \sum_{i=1}^n \beta_i^*(w^{*T}x_i - y_i - \varepsilon)$$

$$= \frac{1}{2}\sum_{i,j=1}^n (\alpha_i^* - \beta_i^*)(\alpha_j^* - \beta_j^*) x_i^T x_j + \sum_{i=1}^n \alpha_i^*\left(\cancel{y_i} - \sum_{j=1}^n (\alpha_j^* - \beta_j^*) x_j^T x_i - \varepsilon\right)$$
$$+ \sum_{i=1}^n \beta_i^*\left(\sum_{j=1}^n (\alpha_j^* - \beta_j^*) x_j^T x_i - \cancel{y_i} - \varepsilon\right)$$

$$= -\frac{1}{2}\sum_{i,j=1}^n (\alpha_i^* - \beta_i^*)(\alpha_j^* - \beta_j^*) x_i^T x_j - \sum_{i=1}^n (\alpha_i^* + \beta_i^*)\varepsilon$$

s.t. $\alpha_i^* \geq 0 \quad \beta_i^* \geq 0. \quad \forall i \qquad \lambda_i^* \geq 0, \ \gamma_i^* \geq 0 \quad \forall i.$

$$\sum_{i=1}^n \alpha_i^* = \sum_{i=1}^n \beta_i^*, \quad \frac{C}{n} = \lambda_i^* + \alpha_i^* = \beta_i^* + \gamma_i^*.$$

由 扫描全能王 扫描创建

Thus, the Dual Optimization Problem is:

$$\max_{\alpha,\beta} \; -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\beta_i)(\alpha_j-\beta_j)x_i^T x_j - \sum_{i=1}^{n}(\alpha_i+\beta_i)\varepsilon$$

s.t. $\alpha_i \geq 0, \; \beta_i \geq 0; \; \lambda_i \geq 0, \; \gamma_i \geq 0, \; \forall i$

$$\sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\beta_i, \quad \frac{C}{n} = \lambda_i + \alpha_i = \beta_i + \gamma_i$$

above constraints equal to: $0 \leq \alpha_i \leq \frac{C}{n}, \; 0 \leq \beta_i \leq \frac{C}{n}, \; \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\beta_i$

c. Let α^*, β^* be dual opt. $\Rightarrow w^* = \sum_{i=1}^{n}(\alpha_i^* - \beta_i^*)x_i$

from complementary slackness,

$$\alpha_i^*(y_i - w^{*T}x_i - b - \varepsilon - \xi_i^+) = 0. \quad \forall i$$

$$\beta_i^*(w^{*T}x_i + b - y_i - \varepsilon - \xi_i^-) = 0$$

consider any i such that $0 < \alpha_i^* < \frac{C}{n}$, since $\alpha_i^* > 0$, we know

$$y_i - w^{*T}x_i - b = \varepsilon + \xi_i^{+*}$$

since $\alpha_i^* < \frac{C}{n}$, we know $\lambda_i > 0. \Rightarrow \xi_i^{+*} = 0$.

With the same reason, we have that for any j s.t. $0 < \beta_j^* < \frac{C}{n}$.

$$w^{*T}x_j + b - y_j = \varepsilon + \xi_j^{-*} \;\; \& \;\; \xi_j^{-*} = 0$$

Solve for b^* $\Rightarrow$ $b^* = y_i - w^{*T}x_i - \varepsilon$ or $b^* = y_j - w^{*T}x_j + \varepsilon$

From the solution of (b) we can easily find that the dual optimization problem is a inner product of $\langle x_i, x_j \rangle$. We can substitute it with $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$, Then SVR solves

$$\max_{\alpha,\beta} \; -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\beta_i)(\alpha_j-\beta_j)k(x_i,x_j) - \sum_{i=1}^{n}(\alpha_i+\beta_i)\varepsilon$$

And SVR is expressed:

$$f(\vec{x}) = \sum_{i=1}^{n}(\alpha_i^* - \beta_i^*)k(x_i, x_j) + b^*$$

where b^* is computed as above.

由 扫描全能王 扫描创建

d. If x_i satisfies: $y_i - w^{*T}x_i - b = \varepsilon + \xi_i^+$

or $w^{*T}x_i + b - y_i = \varepsilon + \xi_i^-$,

we call x_i a support vector.

Therefore, if x_i is not a support vector, then $\alpha_i^* = 0$ or $\beta_i^* = 0$.

Primal:
$$\min_{\vec{w}, b, \xi^+, \xi^-} \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-)$$
$$\text{s.t. } y_i - w^T x_i - b \le \varepsilon + \xi_i^+ \quad \forall i$$
$$w^T x_i + b - y_i \le \varepsilon + \xi_i^- \quad \forall i$$
$$\xi_i^+ \ge 0 \quad \forall i$$
$$\xi_i^- \ge 0. \quad \forall i$$

There are 5 cases:

① if $\left.\begin{array}{l} y_i - w^{*T}x_i - b^* < \varepsilon \\ \text{and } w^{*T}x_i + b - y_i < \varepsilon \end{array}\right\}$, then $\left.\begin{array}{l} \xi_i^{+*} = 0 \\ \text{and } \xi_i^- = 0 \end{array}\right\}$ and x_i is not a support vector

$\xi_i^{-*} = 0$.

② if $y_i - w^{*T}x_i - b^* = \varepsilon$, then $w^{*T}x_i + b^* - y_i \le 0 \le \varepsilon.$, $\xi_i^{+*} = 0$, and

x_i is a support vector

③ if $w^{*T}x_i + b - y_i = \varepsilon$, then $y_i - w^{*T}x_i - b^* \le 0 \le \varepsilon$, $\xi_i^{-*} = 0$, $\xi_i^{+*} = 0$

and x_i is a support vector

④ if $y_i - w^{*T}x_i - b^* > \varepsilon$, then $w^{*T}x_i + b - y_i \le 0 \le \varepsilon$, $\xi_i^{+*} > 0$, $\xi_i^{-*} = 0$, and

x_i is a support vector

⑤ if ~~y_i.~~ $w^{*T}x_i + b - y_i > \varepsilon$, then $y_i - w^{*T}x - b^* < 0 \le \varepsilon$, $\xi_i^{-*} > 0$, $\xi_i^{+*} = 0$

and x_i is a support vector

Since $w^* = \sum_{i=1}^{n}(\alpha_i^* - \beta_i^*)x_i$, if x_i is not a support vector, $\alpha_i^* = \beta_i^* = 0$,

there is no contribution to compute w^* for non-support vector. Also b^* is decided by a support vector,

Thus the final predictor only depend on x_i if it's a support vector, they are a subset of training examples.

由 扫描全能王 扫描创建

EECS 545 Homework 4

Yuan Yin

November 1, 2018

Problem 1

(c)

The result is:

```
MSE for training data is: 19.016860651808518
MSE for test data is: 23.535860253927392
offset b is: 19.429475366627464
```

(d)

The result is:

```
MSE for training data is: 19.249564400341065
MSE for test data is: 26.396905605851654
```

Finding that for kernel ridge regression without offset, the errors both for training and test data are a little larger than kernel ridge regression with offset.

(e)

The code for (c) is:

```
import numpy as np
import scipy.io as sio

# Load the data and preprocessed the data
bodyfat_data = sio.loadmat('bodyfat_data.mat')
x = bodyfat_data['X']
y = bodyfat_data['y']
n,d = x.shape
train_num = 150; m = n - train_num
x_train = x[:train_num,:]; x_test = x[train_num:,:]
```

```
y_train = y[:train_num]; y_test = y[train_num:]
y_train_bar = np.mean(y_train)

# Helper function
def dist2(x,c):
    ndata,dimx = x.shape
    ncenters, dimc = c.shape

    xsum = np.sum(x**2,axis = 1)
    xsum = xsum[:,np.newaxis]
    csum = np.sum(c**2,axis = 1)
    csum = csum[:,np.newaxis]

    n2 = xsum.dot(np.ones([1,ncenters]))+ np.ones([ndata,1]).dot(csum.T)- 2*x.dot(c.T)
    return n2

sigma = 15; lamda = 0.003
kernel = np.exp(-1/2/sigma**2 * dist2(x_train, x_train))
kernel_prime = np.exp(-1/2/sigma**2 * dist2(x_train, x_test))
O1 = 1/train_num * np.ones([train_num, m]); O2 = 1/train_num * np.ones([train_num,
    train_num])
kernel_tilda = kernel - kernel.dot(O2) - O2.dot(kernel) + (O2.dot(kernel)).dot(O2)
kernel_prime_tilda = kernel_prime - kernel.dot(O1) - O2.dot(kernel_prime) +
    (O2.dot(kernel)).dot(O1)

# Kernel ridge regression with offset
y_pre_train = []
for i in range(train_num):
    y_pre_train.append(y_train_bar + (np.mat(y_train - y_train_bar).T
                                    * np.mat(kernel_tilda + train_num * lamda *
                                        np.eye(train_num)).I
                                    * np.mat(kernel_tilda[:,i]).T)[0,0])
y_pre_test = []
for i in range(m):
    y_pre_test.append(y_train_bar + (np.mat(y_train - y_train_bar).T
                                    * np.mat(kernel_tilda + train_num * lamda *
                                        np.eye(train_num)).I
                                    * np.mat(kernel_prime_tilda[:,i]).T)[0,0])

# Compute MSE
MSE_train = sum([(y_train[i] - y_pre_train[i])**2 for i in range(train_num)])/train_num
MSE_test = sum([(y_test[i] - y_pre_test[i])**2 for i in range(n-train_num)])/(n-train_num)
print("MSE for training data is: ", MSE_train[0])
print("MSE for test data is: ", MSE_test[0])
```

```python
# Compute offset b
b = y_train_bar - (np.mat(y_train - y_train_bar).T
                  * np.mat(kernel_tilda + train_num * lamda * np.eye(train_num)).I
                  * np.mat(kernel.dot(O2)[:,0] - ((O2.dot(kernel)).dot(O2))[0,0]).T)[0,0]
print("offset b is: ", b)
```

And the code for (d) is:

```python
import numpy as np
import scipy.io as sio

# Load the data and preprocessed the data
bodyfat_data = sio.loadmat('bodyfat_data.mat')
x = bodyfat_data['X']
y = bodyfat_data['y']
n,d = x.shape
train_num = 150; m = n - train_num
x_train = x[:train_num,:]; x_test = x[train_num:,:]
y_train = y[:train_num]; y_test = y[train_num:]

# Helper function
def dist2(x,c):
    ndata,dimx = x.shape
    ncenters, dimc = c.shape

    xsum = np.sum(x**2,axis = 1)
    xsum = xsum[:,np.newaxis]
    csum = np.sum(c**2,axis = 1)
    csum = csum[:,np.newaxis]

    n2 =  xsum.dot(np.ones([1,ncenters]))+ np.ones([ndata,1]).dot(csum.T)- 2*x.dot(c.T)
    return n2

sigma = 15; lamda = 0.003
kernel = np.exp(-1/2/sigma**2 * dist2(x_train, x_train))
kernel_prime = np.exp(-1/2/sigma**2 * dist2(x_train, x_test))

# Kernel ridge regression without offset
y_pre_train = []
for i in range(train_num):
    y_pre_train.append(np.array(np.mat(kernel + train_num * lamda * np.eye(train_num)).I *
        np.mat(y_train)).T.dot(kernel[i,:].T))
```

```python
y_pre_test = []
for i in range(m):
    y_pre_test.append(np.array(np.mat(kernel + train_num * lamda * np.eye(train_num)).I *
        np.mat(y_train)).T.dot(kernel_prime[:,i]))

# Compute MSE
MSE_train = sum([(y_train[i] - y_pre_train[i])**2 for i in range(train_num)])/train_num
MSE_test = sum([(y_test[i] - y_pre_test[i])**2 for i in range(n-train_num)])/(n-train_num)
print("MSE for training data is: ", MSE_train[0])
print("MSE for test data is: ", MSE_test[0])
```