MACHINE LEARNING
HOMEWORK-3

02/16/16

SAM DAVID CHRIST DOSS PUSHPAM
schristd@andrew.cmu.edu.

1. KERNEL FEATURE MAPPINGS :-

1) $x = (x_1, x_2)^T$, $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$

$$K(x,z) = \phi(x).\phi(z) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T . (z_1^2, \sqrt{2}z_1z_2, z_2^2)^T$$

$$= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$

$$= (x_1z_1 + x_2z_2)^2$$

$$= (x.z)^2$$

2) a) map to feature space and dot product.

$$K(x,z) = \phi(x).\phi(z) = x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$

$\cancel{x_1^2z_1^2 = 4 \text{ operations} \Rightarrow x_1 \times x_1 \times z_1 \times z_1}$

$\cancel{2 \times x_1 \times z_1 \times x_2 \times z_2 = 5 \text{ operations.}}$

$\cancel{x_2^2z_2^2 = 4 \text{ operations.}}$

$$\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$$

$$\left.\begin{array}{r} x_1^2 = 1 \\ \sqrt{2}x_1x_2 = 2 \\ x_2^2 = 1 \end{array}\right\} \Rightarrow 4 \text{ operations for each } \phi$$

dot product.

$$\phi(x)\cdot\phi(x) = \left((x_1^2), (\sqrt{2}x_1x_2), (x_2^2)\right)^T \cdot \left((z_1^2), (\sqrt{2}z_1z_2), (z_2^2)\right)^T$$

$$= \underbrace{\left((x_1^2)(z_1^2) + (\sqrt{2}x_1x_2)(\sqrt{2}z_1z_2) + (x_2^2 z_2^2)\right)}_{5 \text{ operations}}$$

$\phi(x) \Rightarrow 2 \times (4 \text{ for each } \phi) + 5$ operations while dot product

$\Rightarrow 13$ operations

b) $K(x, z) = (x \cdot z)^2$

$x \cdot z = x_1 z_1 + x_2 z_2 \Rightarrow 3$ operations.

$(x \cdot z)^2 \Rightarrow 1$ operations

$\Rightarrow 4$ operations totally.

2) Perceptrons :-

$$y = \varphi\left(\sum_i x_i * w_i + b\right),$$

$$\varphi(z) = \begin{cases} 0 & \text{if } z \le 0 \\ 1 & \text{otherwise} \end{cases}$$

1) AND operation

x_1	x_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

2) w_1 and w_2 for AND.

$w_1 = 1$
$w_2 = 1$
$b = -1$

3) OR:-

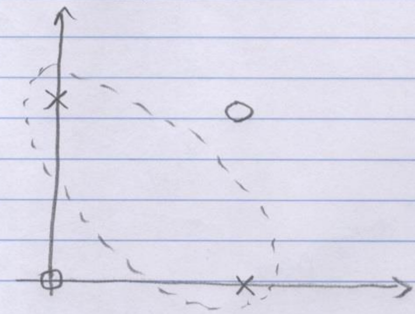
x_1	x_2	OR
0	0	0
0	1	1
1	0	1
1	1	1

4) $w_1 = 1$, $w_2 = 1$, $b = 0$

5) XOR :-

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

6) Single layer perceptron can't be used to create logical XOR.



The XOR function is not linearly separable, which is possible in AND, OR functions. The above graph shows the plot that linear line cannot be drawn to separate (1,1), (0,0) from (0,1), (1,0). So it is not possible to find a values for w_1, w_2, b in the line equation $w_1 x_1 + w_2 x_2 + b$

3. Regression theory :-

3.1 Linear regression :-

1) $f(x) = w^T x$

a) Squared loss, $J(w)$ associated with $f(x)$

$$J(w) = \frac{1}{2}\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$$

b) Partial derivative w.r.t. w^k.

$$\frac{\partial J(w)}{\partial w^k} = \frac{\partial}{\partial w^k} \frac{1}{2}\sum_i (y_i - f(x_i))^2 = \frac{2}{2}\sum_i (y_i - f(x_i)) \frac{\partial}{\partial w_k} f(x_i)$$

$$\frac{\partial}{\omega_k} f(x_i) = \frac{\partial}{\partial \omega_k} \omega^T x_i \Rightarrow (x_i)^k.$$

$$\Rightarrow \frac{\partial}{\partial \omega^k} J(\omega) = \frac{2}{2} \sum_i (y_i - f(x_i))(x_i)^k = \sum_i (y_i - f(x_i)) x_i^k$$

c) update rule:-

$$\omega^k_{new} = \omega^k + \alpha \cdot \frac{\partial}{\partial \omega^k} J(\omega)$$

$$= \omega^k + \alpha \cdot \left(\frac{2}{2} \sum_i (y_i - f(x_i)) x_i^k \right)$$

$$\omega^k_{new} = \omega^k + \alpha \sum_i (y_i - f(x_i)) x_i^k$$

~~where, α~~ with, $\alpha < 0$

2) Maximum likelihood:-

a) $y_i \sim N(\omega^T x_i, \sigma^2)$

Given that y_i is iid in normal distribution.

$$L(\omega; x, y) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - f(x_i))^2}{2\sigma^2}}$$

$\Rightarrow$ This is due to the condition that y_i is iid,

b) Log of conditional likelihood :-

$$l(\omega; x, y) = \log(L(\omega; x, y))$$

$$= \log\left(\prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(y_i - f(x_i))^2}{2\sigma^2}\right)}\right)$$

$$l(\omega; x, y) = n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{\sum_i (y_i - f(x_i))^2}{2\sigma^2}$$

To maximize $l(\omega; x, y)$, Take derivative w.r.t. ω^k.

$$\frac{\partial}{\partial \omega^k} l(\omega; x, y) = -\frac{1}{2\sigma^2} \frac{\partial}{\partial \omega_k} \sum_i (y_i - f(x_i))^2$$

$$= -\frac{1}{2\sigma^2} 2 \sum_i \left(y_i - f(x_i) \frac{\partial f(x_i)}{\partial \omega_k}\right)$$

$$\frac{\partial}{\partial \omega_k} f(x_i) = x_i^k$$

$$= -\frac{1}{\sigma^2} \sum_i \left((y_i - f(x_i))\, x_i^k\right)$$

c) $$l(\omega; x, y) = -\frac{\sum_i (y_i - f(x_i))^2}{2\sigma^2} + n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)$$

Maximizing l is same as minimizing $-l$

So, minimizing $-l$ is written as.

$$(-l) = \sum_i \frac{(y_i - f(x_i))^2}{2\sigma^2} - n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)$$

The second term is constant. This formula will give the same result as minimizing in (1b) which is,

$$\text{minimizing } \frac{1}{2} \sum_i (y_i - f(x_i))^2$$

So, maximizing $l(w; x, y)$ is same as minimizing least square error.

3) $y = f(x) + \epsilon$, ϵ has mean 0 and variance σ^2

a) Expected loss under squared error at some test point x.

$$E_{D,\epsilon}((y - \hat{f})^2) \Rightarrow E_{D,\epsilon}((y - \hat{f})^2),$$

where $y = f(x) + \epsilon$

The expected loss is taken over the dataset and the noise.

$y = f(x) + \epsilon$, $\hat{f} = h_D(x)$

$\to h_D$ is the linear regressor trained using D.

b) $MSE = \underset{D,\epsilon}{E}\left((y-\hat{f})^2\right)$

$$= \underset{D,\epsilon}{E}\left((y - E(\hat{f}) + E(\hat{f}) - \hat{f})^2\right)$$

$$= \underset{D,\epsilon}{E}\left[(y-E(\hat{f}))^2 + (E(\hat{f})-\hat{f})^2 + 2(y-E(\hat{f}))(E(\hat{f})-\hat{f})\right]$$

$$= \underset{D,\epsilon}{E}\left((y-E(\hat{f}))^2\right) + \underset{D,\epsilon}{E}\left((E(\hat{f})-\hat{f})^2\right) + 2\underset{D,\epsilon}{E}\left\{(y-E(\hat{f}))(E(\hat{f})-\hat{f})\right\}$$

$$= \underset{D,\epsilon}{E}\left((y-E(\hat{f}))^2\right) + \underset{D,\epsilon}{E}\left((E(\hat{f})-\hat{f})^2\right) + 2(y-E(\hat{f}))(E(\hat{f})-E(\hat{f}))$$

$$= \underset{D,\epsilon}{E}\left((f(x)+\epsilon - E(\hat{f}))^2\right) + \underset{D,\epsilon}{E}\left((E(\hat{f})-\hat{f})^2\right) + 0$$

$$= \underset{D,\epsilon}{E}\left((f(x)-E(\hat{f}))^2\right) + \underset{D,\epsilon}{E}(\epsilon^2) - 2\underset{D,\epsilon}{E}\left\{(f(x)+\epsilon)(\epsilon)\right\} + \underset{D,\epsilon}{E}\left((E(\hat{f})-\hat{f})^2\right)$$

$$= \underset{D,\epsilon}{E}\left((f(x)-E(\hat{f}))^2\right) + \sigma^2 + \underset{D,\epsilon}{E}\left((E(\hat{f}))-\hat{f})^2\right) - 2\left(E(f(x))+E(\epsilon)\right)(E(\epsilon))$$

$\Rightarrow$ Since, $E(\epsilon) = 0 \Rightarrow$ mean is 0, last term becomes 0.

$$MSE = E_{D,\epsilon}\left(\hat{f}(x) - E(\hat{f}(x))\right)^2 + E\left[\left(E(\hat{f}) - f\right)^2\right] + \sigma^2$$

$$= \text{Bias}^2 + \text{Variance} + \sigma^2$$

3.2 Regularization

1) a) $$L = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - w^T x_i\right)^2 + \frac{\lambda}{2}\|w\|^2$$

$$\frac{\partial L}{\partial w_k} = \frac{2}{2}\sum_{i=1}^{n}(y_i - w^T x_i)\frac{\partial(w^T x_i)}{\partial w_k} + \frac{\lambda}{2}\frac{\partial}{\partial w_k}\|w\|^2$$

$$= \sum_{i=1}^{n}(y_i - w^T x_i)\, x_i^k + \lambda w_k$$

b) The algorithm(ii) is most likely to give sparse w. The value of w is normalized in that case, so it may suppress the value to 0 completely.

with the normalgation factor actually suppresing the value of w^k, (ii) tends to give sparse w.

4. Programming logistic regression.

4.1 Logistic regression.

1) Logistic function $f(x;\omega)$

$$f(x;\omega) = \frac{1}{1+e^{(-\omega^T x)}}$$

2). Conditional likelihood.

$$L(\omega; x, y) = \begin{cases} f(x;\omega), & \text{if } y=1 \\ 1-f(x;\omega), & \text{if } y=0 \end{cases}$$

3). Log conditional likelihood.

Log Likelihood on One example.

$$\Rightarrow \begin{cases} \log(f(x;\omega)), & \text{if } y=1 \\ \log(1-f(x;\omega)), & \text{if } y=0 \end{cases}$$

4). Derivative of log likelihood

$$\frac{\partial}{\partial \omega^k} \ell(\omega; x, y) \Rightarrow \begin{cases} \frac{\partial}{\partial \omega_k} \log(f(x;\omega)), & \text{if } y=1 \\ \frac{\partial}{\partial \omega_k} \log(1-f(x;\omega)), & \text{if } y=0 \end{cases}$$

$$\frac{\partial}{\partial \omega_k} \ell(\omega; x, y) = \begin{cases} \frac{1}{f} \frac{\partial}{\partial \omega_k} f(x;\omega), & \text{if } y = 1 \\ \frac{1}{1-f} \left(\frac{-\partial}{\partial \omega_k} f(x;\omega) \right), & \text{if } y = 0 \end{cases}$$

$$f = \frac{1}{1+e^{-\omega^T x}}, \quad 1 - f = \frac{e^{-\omega^T x}}{1+e^{-\omega^T x}}$$

$$\frac{\partial}{\partial \omega_k} f \Rightarrow (-1)\left(1+e^{-\omega^T x}\right)^{-2} \frac{\partial}{\partial \omega_k}\left(e^{-\omega^T x}\right)$$

$$\Rightarrow (-1)\left(1+e^{-\omega^T x}\right)^{-2}\left(e^{-\omega^T x}\right)\left(-x^k\right)$$

$$\Rightarrow \frac{1}{\left(1+e^{-\omega^T x}\right)} \cdot \frac{e^{-\omega^T x}}{\left(1+e^{-\omega^T x}\right)} x^k$$

$$\frac{\partial}{\partial \omega_k}(f) \Rightarrow \left(f(x;\omega)\right)\left(1-f(x;\omega)\right) x^k$$

$$\frac{\partial}{\partial \omega_k} \ell(\omega; x, y) = \begin{cases} \frac{1}{f} \cdot f(1-f)\, x^k, & \text{if } y = 1 \\ \left(\frac{1}{1-f}\right)\left(-f(1-f)\, x^k\right), & \text{if } y = 0 \end{cases}$$

$$= \begin{cases} (1-f)\, x^k, & \text{if } y = 1 \\ -f x^k, & \text{if } y = 0 \end{cases}$$

This can also be written as,

$$\frac{\partial}{\partial \omega_k} \ell(\omega; x, y) = \sum_{i=1}^{n} (y_i - f)\, x^k$$

5) The update rule is written as,

$$w^{k}_{new} = w^{k} - \alpha \frac{\partial}{\partial w_k} \ell(w; x, y)$$

$$w^{k}_{new} = w^{k} - \alpha \sum_{i=1}^{n} + (y_i - f(w; x, y))\, x_i^k$$

6) $$\arg\min_{w} \; -\ell(w; x, y) + \frac{\lambda}{2}\|w\|^2$$

Taking derivative w.r.t. w_k gives

$$\Rightarrow -\frac{\partial}{\partial w_k} \ell(w; x, y) + \frac{\partial}{\partial w_k}\left(\frac{\lambda}{2}\|w\|^2\right)$$

$$\Rightarrow -(y_i - f(w; x, y))\, x_i^k + \lambda w_k$$

4.3)

1) For Unregularized LR,

Training accuracy :- 99.1575 %

Testing accuracy :- 98.607 %

For regularized LR,

Training accuracy :- 99.2236 %

Testing accuracy :- 98.607 %

Training accuracy increased slightly in, regularized LR, Whereas the testing accuracy remains the same. The gap between training test error becomes wider as a result.

We would expect a ~~decrease~~ ~~to~~ increase in testing accuracy as well. This might be due to low value of the regularization parameter, that it doesnot affect the parameter values significantly.

5) Programming Kernel perceptron :-

5.2)

4. Perceptron :-

Training accuracy = 98.8932 %

Testing accuracy = 98.5075 %

Kernel perceptron:

Training accuracy : 99.7605%

Testing accuracy : 99.602%

Kernel perceptron has higher accuracy in both training and testing set. This is because the kernel perceptron actually works in higher dimensional space implicitly, which results in better classification.

4.3)

2)