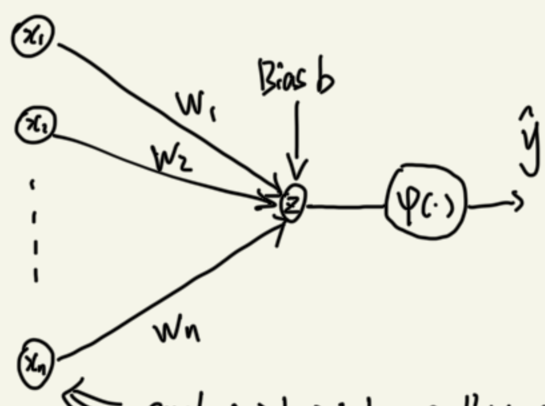
陈羿乔 YiqiaoChen 2024/10/1

① Perceptron Model



Notation:

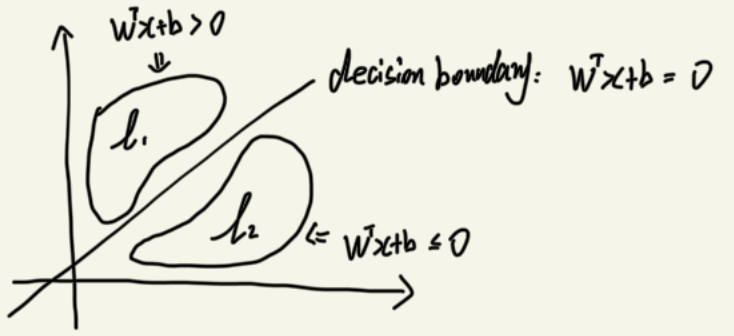
I. w: weight, $w \in \mathbb{R}^m$
II. b: bias, $b \in \mathbb{R}$
III. x: input vector, $x \in \mathbb{R}^m$
IV. y: label for each x, $y \in \mathbb{R}$
V. $\hat{y}$: model predict label, $\hat{y} \in \mathbb{R}$
VI. $\varphi(\cdot)$: activation function, $\varphi(\cdot): \mathbb{R}^m \to \mathbb{R}$

⇐ each input x has a "+1 / -1" label named y.

i. $$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = \sum_{i=1}^{n} w_ix_i + b = \langle w, x\rangle = w^Tx$$ ⇐ Linear combination

ii. $\hat{y} = \varphi(z)$, $\varphi(\cdot): x \mapsto \mathrm{sgn}(x)$ ⇐ linear to non-linear

② Perceptron Training



i. weight update for perceptron

perceptron only update its weight when misclassify

case 1: correctly classified. we have
$$w(n+1) = w(n)$$

case 2: incorrectly classified. i.e. $y(n+1)\,\underbrace{w(n)^Tx(n+1)}_{\hat{y}(n+1)} < 0$

$$w(n+1) = w(n) + y(n+1)x(n+1)$$

we first initialize the model with
$w(0)=0$, $b(0)=0$
and then we start input
$x(1)$ and its lable $y(1)$
we calculate and compared the result
if $y(1)w(0)^Tx(1) < 0$
$w(1) = w(0) + y(1)x(1)$
and then input the next, $x(2)$

e.g. consider input same data for $w(n+1)$

$$y(n)w(n+1)^Tx(n)$$
$$= y(n)[w(n) + y(n)x(n)]^Tx(n)$$
$$= y(n)w(n)^Tx(n) + y(n)^2\|x(n)\|^2$$
$$= y(n)w(n)^Tx(n) + \|x(n)\|^2$$

always positive,
which means the negative component
$y(n)w(n)^Tx(n)$ is turning to more positive.

P.S.1 where notation (n) means n^{th} input, w, b start from $w(0), b(0)$ ⇐ initial weight, bias

P.S.2 sometimes we add learning rate η for update.

i.e. if $y(n)w(n)^Tx(n) < 0$ {

$w(n+1) = w(n) + \eta\, y(n)x(n)$;

}

where $\eta \in [0, 1]$, which control the speed that model trains.

ii. Perceptron convergence

In order to simplify the derivation, we add the bias term b and its coefficient "+1" to the w and x first place, respectively.

i.e. $\hat{y} = \mathrm{sgn}(w^T x)$

$w = [b \ w_1 \ w_2 \cdots w_n]^T$

$x = [1 \ x_1 \ x_2 \cdots x_n]^T$

$\Rightarrow$

① Hypothesis 1: All training datas have bounded Euclidean norms

i.e. $\| x(t) \|_{L^2} \leq \varepsilon$ for all $t \in \mathcal{X}$

② Hypothesis 2: Linear classifier with weight W_* exists

i.e. $y(t)\, W_*^T x(t) \geq \gamma$ for all $t \in \mathcal{X}$

and the classifier just right before W_* is $w(n)$

③ Aim: using the constraint for $\cos(\cdot) \leq 1$.

set up $\cos(W_*, w(n))$ to find the boundary

$$\frac{W_*^T w(n)}{\|W_*\| \, \|w(n)\|} \leq 1$$

a) Part 1.

we can first decomposition $w(n)$ using the "update rule"

$$w(n) = w(n-1) + \alpha(n) y(n) x(n)$$

$$\alpha(n) = \begin{cases} 1, & \text{if misclassify at } n^{th} \text{ input} \\ 0, & \text{otherwise} \end{cases}$$

$$= w(n-2) + \alpha(n-1) y(n-1) x(n-1) + \alpha(n) y(n) x(n)$$

$$\vdots$$

$w(0) = 0$

$$= w(n-n) + \alpha(1) y(1) x(1) + \alpha(2) y(2) x(2) + \cdots + \alpha(n) y(n) x(n)$$

$$= \sum_{i=1}^{n} \alpha(i) y(i) x(i)$$

thus we can rewrite the inner product for $\langle W_*, W(n) \rangle$

$$W_*^T w(n) = W_*^T \left(\sum_{i=1}^{n} \alpha(i) y(i) x(i) \right)$$

$$= \sum_{i=1}^{n} \alpha(i)\, y(i) W_*^T x(i)$$

recall the hypothesis 2, we have $\min_i \{ y(i) W_*^T x(i) \} = \gamma$

not every input cause update, let's just assume k's update exist

$$\geq \sum_{i=1}^{n} \alpha(i) \gamma$$

$$\geq k\gamma \iff W_*^T w(n) \geq k\gamma$$

At the same time we know

$W_*^T w(n) \geq k\gamma \Rightarrow$

thus we have

$$[W_*^T w(n)]^2 \geq k^2 \gamma^2$$

Using the Cauchy-Schwarz Inequality,

$$\|W_*\|^2 \, \|W(n)\|^2 \geq [W_*^T w(n)]^2 \geq k^2\gamma^2$$

$$\Downarrow$$

$$\|w(n)\|^2 \geq \frac{k^2 \gamma^2}{\|W_*\|^2} \text{, lower bound}$$

b) Part 2.

Calculate $\| W(n) \|^2$

$$= \| W(n-1) + \alpha(n) y(n) x(n) \|^2$$

$$= \| W(n-1) \|^2 + \| \alpha(n) x(n) \|^2 + 2 \alpha(n) y(n) W(n-1)^T x(n)$$

if no update at n^{th} input: $\alpha(n) = 0$

if update at n^{th} input: $y(n) W(n-1)^T x(n) < 0$

$$\leq \| W(n-1) \|^2 + \| \alpha(n) x(n) \|^2$$

$$\leq \| W(n-2) \|^2 + \| \alpha(n-1) x(n-1) \|^2 + \| \alpha(n) x(n) \|^2$$

$$\leq \sum_{i=1}^{n} \| \alpha(n-1) x(n-1) \|^2$$

recall the hypothesis 1: $\min_i \{ \| x(i) \| \} = \varepsilon$

using the same assume as part 1: k^{th} update

$$\leq k\varepsilon^2$$

c) Conclusion

a. conclusion 1: upper / lower bound for $\|W(n)\|^2$

$$\frac{k^2\gamma^2}{\|W_*\|^2} \leq \|W_n\|^2 \leq k\varepsilon^2$$

b) conclusion 2: update time k

$$\cos(W_*, W(n)) = \frac{W_*^T W(n)}{\|W_*\|\,\|W(n)\|} \geq \frac{k\gamma}{\|W_*\|\,\|W(n)\|} \geq \frac{k\gamma}{\|W_*\|\sqrt{k\varepsilon^2}}$$

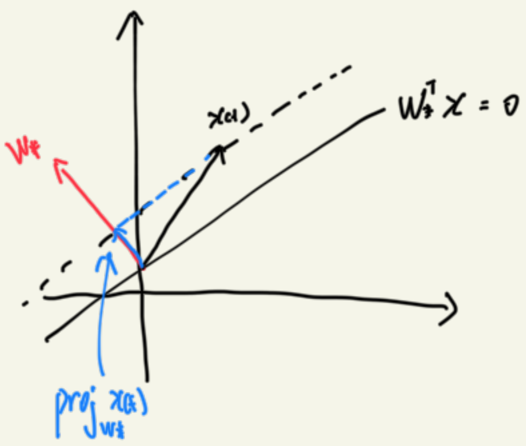
Part 1

Part 2

that is $1 \geq \frac{k\gamma}{\|W_*\|\sqrt{k\varepsilon^2}} \Longleftrightarrow k \leq \frac{\varepsilon^2\|W_*\|^2}{\gamma^2}$

d) Advanced derivation

Set $x(t)$ be the data sample most close to decision boundary (i.e. $y(t)W_*^T x(t) = \gamma$)



Using projection and norm to calculate distance is invented by my own. inspired by the lecture notes from MIT.

$$\left\| \text{proj}_{W_*} x(t) \right\| = \left\| \frac{x(t)^T W_*}{W_*^T W_*} \cdot W_* \right\|$$

$y(t) W_*^T x(t) = \gamma$

$|W_*^T x(t)| = \gamma$

$$= \frac{|W_*^T x(t)|}{\|W_*\|^2} \cdot \|W_*\|$$

$$d_{min} = \frac{\gamma}{\|W_*\|}$$

$\Leftarrow$ the min distance for the data set to the decision boundary

rewrite the $k \leq \frac{\varepsilon^2\|W_*\|^2}{\gamma^2}$, to

$$k \leq \frac{\varepsilon^2}{d_{min}^2}$$

$\leftarrow$ the more small the geometric margin that separates the training set, the more difficult for the training.

③ Batch Perceptron : changing the size we input

i. Premise : We input a set of sample x named $\mathcal{H} = \{x(1), x(2) \dots x(n)\}$
Instead of input one by one like we used to do.
And for the dataset $\mathcal{H}$, a batch of misclassified samples $\mathcal{X}$ is used to compute the adjustment. and $\mathcal{X} \in \mathcal{H}$.

ii. Error : For those misclassified samples, we compute the total error

$$\overset{1\times1}{J(w)} = \sum_{x(t)\in\mathcal{X}} (-y(t)\, w^T x(t))$$

and how w changes effect to the total error

$$\overset{m\times1}{\nabla J(w)} = \frac{\partial J(w)}{\partial w} = \sum (-y(t)\, x(t))$$

iii. Update:

(Method of Gradient Descent)

$$W' := W - \underline{\eta \nabla J(w)}$$

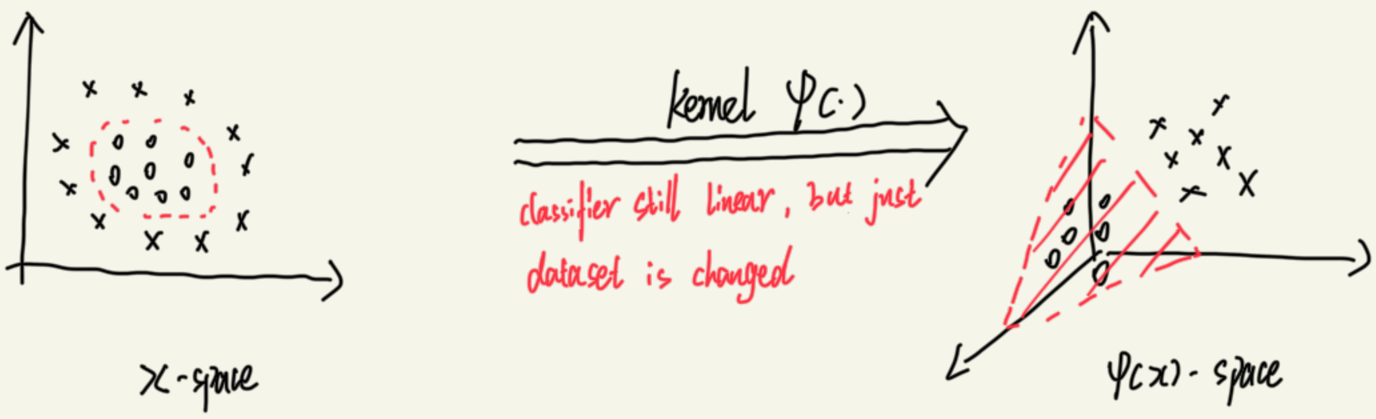
$$= W + \underline{\eta \Big(\sum_{x(t) \in \mathcal{X}} y(t) x(t) \Big)}$$

Accumulate all updates one by one into a single update

Recall the one by one input case.

$$W' = W + \eta\, y(n)\, x(n)$$

④ Kernel Method



kernel: Mapping an original feature vector x to an expanded version $\varphi(x)$

information in x is expanded

i. Feel hard to explicitly compute $\varphi(\cdot)$?

Solution: Noticed that the main calculate in learning model is "inner product".

$$\boxed{\begin{array}{l} \text{use } k(x,z) \text{ to replace } \varphi(x)^T\varphi(z) \\ \text{define } k(\cdot,\cdot) \text{ to skip define } \varphi(\cdot) \end{array}}$$

example: define $\varphi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$

calculate $\varphi(x)^T\varphi(z)$, $x = [x_1, x_2]^T$, $z = [z_1, z_2]^T$

$$\varphi(x)^T\varphi(z) = [x_1^2, \sqrt{2}x_1x_2, x_2^2] \begin{bmatrix} z_1^2 \\ \sqrt{2}z_1z_2 \\ z_2^2 \end{bmatrix}$$

$$= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$

insted we can define a function have same result

$$k(x,z) = (x^Tz)^2 = x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$

$$= \varphi(x)^T\varphi(z)$$

ii. Use kernel method in perceptron

Rewrite the weight in "dec-type".

step1: Initialize weight $w = 0$, $b = 0$

$$W(n-1) = W(n-2) + \alpha(n-1)\, y(n-1)\, \varphi(x(n-1))$$

$\vdots$ n-1

step2: At i-th input, we have weight represent by

Step2: At the n^{th} input, we have weight represent by

$$= \sum_{i=1} \alpha(i) y(i) \varphi(x(i))$$

$$\boxed{W(n-1) = \sum_{i=1}^{n-1} \alpha(i) y(i) \varphi(x(i))}$$

Step3: Calculate output before activated, using kernel

$$z(n) = W(n-1)^T \varphi(x(n)) + b$$

$$= \left[\sum_{i=1}^{n-1} \alpha(i) y(i) \varphi(x(i)) \right]^T \varphi(x(n)) + b$$

inner product

$$= \sum_{i=1}^{n-1} \alpha(i) y(i) \left[\varphi(x(i))^T \varphi(x(n)) \right] + b$$

$$= \sum_{i=1}^{n-1} \alpha(i) y(i) \, k(x(i), x(n)) + b$$

Step4: Mark the update. Execute update in next iteration instead real changing the value of $W(n)$

we simply mark $\alpha(n) = 1$, if the n^{th} input is misclassify

Then the update will take effect during the next calculation output

i.e. in $n+1^{th}$ input

$$z = \left[\sum_{i=1}^{n} \alpha(i) y(i) \varphi(x(i)) \right]^T x(n+1)$$

$$= \left[\sum_{i=1}^{n-1} \alpha(i) y(i) \varphi(x(i)) + \alpha(n) y(n) \varphi(x(n)) \right] x(n+1)$$

update is not execute when I identify a misclassification (i.e. if ...) but execute when calculate the output before the activating function. $\Rightarrow$

$$W(n) = W(n-1) + \alpha(n) y(n) \varphi(x(n))$$

$$= \sum_{i=1}^{n-1} \alpha(i) y(i) \, k(x(i), x(n+1)) + \alpha(n) y(n) \, k(x(n), x(n+1))$$

$$= \sum_{i=1}^{n} \alpha(i) y(i) \, k(x(i), x(n+1))$$