$K = X \cdot X^T$ linear kernel

$K_{ij} = K_{ji} = \langle x_j, x_i \rangle = x_j \cdot x_i^T$

$= x_i \cdot x_j^T$

linear sim
linear corelation.

- primary

$h(x) = \boxed{x} \cdot w \ldots$ → primary form (data)

- dual form has input $sim(x_i, x_j) = K_{ij}$, not the data (x)

$h(x) = \sum \ldots \langle x_j \cdot x_i \rangle$

$sim(x_i, x_j) = K_{ij}$

example x_i = patient in hospital.

$h(x) =$ function (patient data
blood pressure,
age,
symptoms)

dual:
$h(x) =$ fraction of similarity of (x, other patients)

Kernel Trick

- $h(x) \Rightarrow$ dual form $\quad h(x) = \text{function}(k_{ij})$
  $\langle x_i \cdot x_j \rangle = \text{sim}(x_i, x_j)$
- (Th) h works with any valid kernel
  (not just linear kernel)

- plug in a domain-specific/sim-designed kernel (valid)
  ex gaussian kernel $k_{ij} = e^{\|x_i - x_j\|^2 / \beta}$ (prove valid)
- use the classifier $h()$ with non-linear kernel.

- effect: $h(x)$ linear $\Rightarrow$ dual $h(x) = \text{function}(\overset{\text{non-lin}}{k})$
  classifier
  (ex. regression, SVM)
  can do non-linear separation!

Lecture 7/16: - KNN classifier - distance/similarities.

- Kernelization / duality of linear classifiers

- HW5: SVM + kernels demo Friday 7/18 Caleb

- Projects data update

  - this week: basic setup

    not nec. yours

    ALG + data + result / not working

    - simple (online)
    - sample
    - not good

    - concise writeup

- next week: feedback + direction

  - modify ALG
  - more data
  - diff data
  - computation / scale

  - debug
  - evaluation
  - other idea.

Fundamental ML: similarity (x_i, x_j) between 2 datapoints

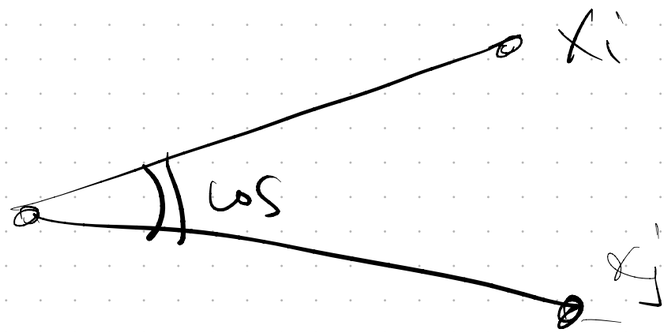
$\sim$ distance (x_i, x_j) opposite

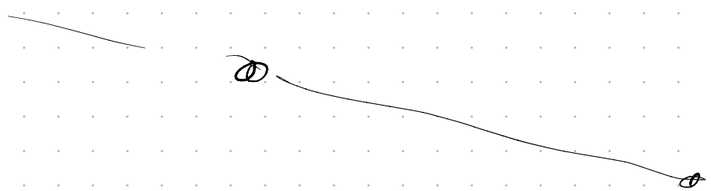
(Th) correct/exact $\text{sim}(x_i, x_j) \equiv$ correct classification/ML

- default sim with linear algebra $= \langle x_i x_j \rangle = \sum_{d=1}^{D} x_i^d \cdot x_j^d$

$\equiv$ linear kernel $K_{ij} = [X \cdot X^T]_{ij} = \langle x_i x_j \rangle$

- $\text{cosine}(x_i, x_j) = \dfrac{\langle x_i x_j \rangle}{\sqrt{\|x_i\| \, \|x_j\|}} = \dfrac{\text{dot product}}{\text{normalized by length-gain}}$





- Euclidian distance $\|x_i - x_j\|^2 = \sum_{d=1}^{D} (x_i^d - x_j^d)^2$

  (Norm 2 distance)

- Manhattan distance $\|x_i - x_j\|_1 = \sum_{d=1}^{D} |x_i^d - x_j^d|$

  Norm 1 distance

- Norm p distance $\|x_i - x_j\|^p = \sum_{d=1}^{D} |x_i^d - x_j^d|^p$

- Mahalanobis distance $\text{mahal}(x_i, x_j) = \sqrt{(x_i - x_j) \sum^{-1} (x_i - x_j)^T}$

  $\sum^{-1}$ COVAR INVERSE

  - if $\sum_{\text{COVAR}}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ → unit variance on all feat
    → 0 covar between feat
    $\Rightarrow$ Euclid distance $\sqrt{(x_i - x_j)(x_i - x_j)^T}$

- If $\Sigma_{COVAR} = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \sigma_D^2 \end{bmatrix}$ → no covar between features

⇒ weighted / normalized Euclidian Distance
all features matter same

$$\sqrt{(x_i - x_j)\begin{bmatrix} \frac{1}{\sigma_1^2} & & 0 \\ & \frac{1}{\sigma_2^2} & \\ 0 & & \frac{1}{\sigma_D^2} \end{bmatrix}(x_i - x_j)^T}$$

- Euclidian dist ⇒ similarity

Gaussian / RBF kernel

$$k_{ij} = sim(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2}$$

σ = scaler

- max $k_{ij} = e^0 = 1$ when $x_i = x_j$
- min $k_{ij} = e^{-\infty} = 0$

- **Edit distance** (Hamming)

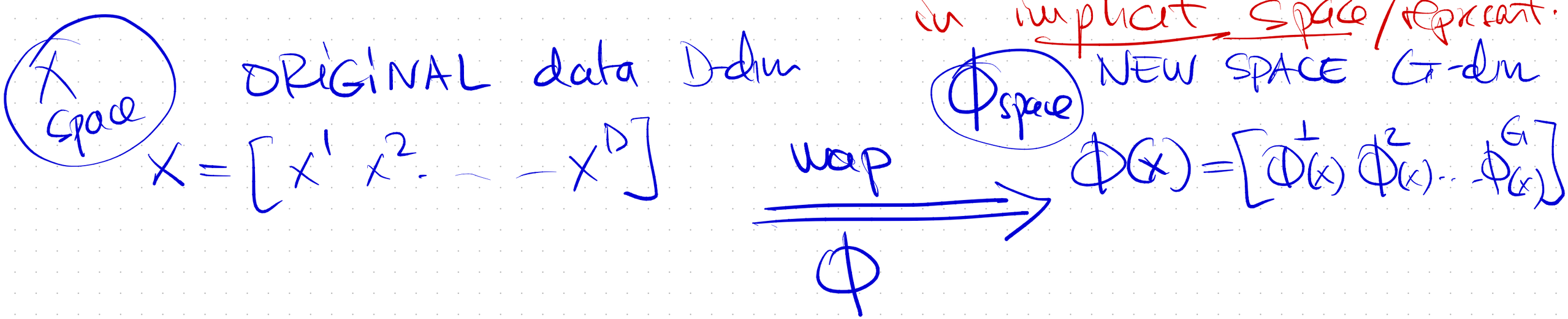
$$ham(x_i, x_j) = \sum_{d=1}^{D} 1[x_i^d \neq x_j^d]$$

$= \#$ of different feat values.

different $\equiv |x_i^d - x_j^d| > \varepsilon$

KERNEL = SIMILARITY $\quad K = [K_{ij}]_{\substack{i=1:N \\ j=1:N}}$ kernel matrix $N \times N$

- not any similarity (domain specific) is a valid kernel (math)!
- what are valid kernels? K = always linear but in implicit space/repr.



K = valid kernel $\Leftrightarrow$ $K = \phi(x) \cdot \phi(x)^T$

$K_{ij} = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

linear in ϕ-space

- linear kernel $K = x \cdot x^T$ uses no ϕ, or $\phi(x) = x$

Kernel Trick in ML

- dont use ϕ explicit, only need existance ($\phi(x)$) theory (do not compute $\phi()$ know map)

- primal ALG $h(x) = f(x)$ $\Longrightarrow$ dual form $h(x) = f(K)$

then we can use $h()$ algorithm with any kernel. (Th, proves math)

k examples $K_{ij} = (\langle x_i x_j \rangle)^2$ valid kernel ✓

explicit ϕ for this toy example

original $x = [x^1 \quad x^2]$ (2-dim) $\longrightarrow$ $\phi(x) = [(x^1)^2, (x^2)^2, x^1 \cdot x^2, x^2 x^1]$ (4-dim)

$\langle \phi(x_i) \cdot \phi(x_j) \rangle$ = linear similarity in ϕ space

$= [(x_i^1)^2 \; (x_i^2)^2 \; (x_i^1 x_i^2) \; (x_i^2 x_i^1)] \times$

$[(x_j)^2 \; (x_j^2)^2 \; (x_j^1 x_j^2) \; (x_j^2 x_j^1)]$

$= (x_i^1 \cdot x_j^1)^2 + (x_i^2 x_j^2)^2 + 2 x_i^1 x_j^1 x_i^2 x_j^2$

$= (x_i^1 x_j^1 + x_i^2 x_j^2)^2$

$= (\langle x_i x_j \rangle)^2$

calculate w/out ϕ

$(x_i x_j^T)^2$

$K_{ij} = (\langle x_i x_j \rangle + 1)^2$ valid kernel.

calculate $k_{ij} = f(x_i, x_j)$ without ϕ

exercise $k_{ij} = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

original $[x^1, x^2] \to \phi(x) = [(x^1)^2, (x^2)^2, \sqrt{2}x^1, \sqrt{2}x^2, \sqrt{2}x^1x^2, 1]$ 6-dim

---

in general: polynomial kernel $k_{ij} = (\langle x_i \cdot x_j \rangle + c)^p$

valid $\Leftrightarrow \exists \phi(x)$ map such that $k_{ij} = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

practice: we donot calculate/explicit use ϕ

Gaussian kernel

$K_{ij} = e^{-1/2 \|x_i - x_j\|^2}$?? $\sum_{t=0}^{\infty} \frac{(x_i x_j^T)^t}{t!} e^{-1/2\|x_i\|^2} e^{-\frac{1}{2}\|x_j\|^2}$

VALID

$K = \Phi_{(x)} \Phi_{(x)}^T$

dot product $\langle \Phi(x_i) \cdot \Phi(x_j) \rangle$

Φ infinite dimensionality

calculate w/out Φ: $e^{-1/2 \|x_i - x_j\|^2}$

Linear Algorithms primal $\Rightarrow$ dual $h(x) = \text{function}(xx^T) = \text{function}(K)$

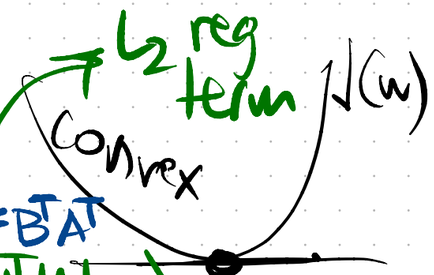
- ✓ SVM
- ✓ Linear Reg
- PCA
- ✓ Perceptron
- KNN with kernels

to enable usage of K trick

Recall Linear Regression Normal Eq with L_2-penalty (May 14 Lecture)

intuition $J(w) = \frac{1}{2}(Xw-Y)^T(Xw-Y)$ convex in w

$\Rightarrow \frac{\partial J}{\partial w} = 0$ when $J(w)$ is minimum.



Q?

$\frac{\partial J}{\partial w} = \frac{1}{2}\frac{\partial}{\partial w}(Xw-Y)^T(Xw-Y)$

$= \frac{1}{2}\frac{\partial}{\partial w}(w^TX^TXw - w^TX^TY - Y^TXw - Y^TY) + \frac{1}{2}\frac{\partial}{\partial w}w^Tw\cdot\lambda$

1×1 dim (1×D)(D×N)(N×1)

same transposed

$(AB)^T = B^TA^T$

$+\frac{1}{2}\frac{\partial}{\partial w}w^Tw\cdot\lambda$

$= \frac{1}{2}\frac{\partial}{\partial w}(w^TX^TXw - 2w^TX^TY - Y^TY) + \frac{1}{2}\frac{\partial}{\partial w}w^Tw\cdot\lambda$

Y^TY: $\partial = 0$?

$= \frac{1}{2}\frac{\partial}{\partial w}w^TX^TXw - \frac{\partial}{\partial w}w^TX^TY + \frac{1}{2}\frac{\partial}{\partial w}w^Tw\cdot\lambda$

pretend is 1-dim(w)

pretend 1-dim

$= \frac{1}{2}2X^TXw - X^TY + \lambda I\cdot w$

λI · w

Id. matrix

$= X^TXw - X^TY + \lambda Iw$

want $\equiv 0$

$X^TXw = X^TY$ → sq DxD, pos definite (pos eig val)

invert-left by $(X^TX)^{-1}$

$\Rightarrow (X^TX+\lambda I)w = X^TY$

$w = (X^TX)^{-1}X^TY$ NORMAL EQ (closed form)

Best (Th) w for min sq errs of reg. line

$w = (X^TX+\lambda I)^{-1}\cdot X^TY$

L_2-reg

Dualize Regression.

Primal: compute w coef

$$\left[X^T X + \lambda \underbrace{I_D}_{\text{id } D\times D}\right] w = X^T y$$

Dual: $w = X^T \cdot \alpha$

var α : = lin comb (X^T) with coef α.

$$\underset{N\times 1}{\alpha} = \left[\underset{N\times N}{X X^T} + \lambda \underbrace{I_N}_{\text{id } N\times N}\right]^{-1} \underset{N\times 1}{y}$$

α_i = coef for x_i

Prof that $w = X^T \alpha$

check the calculation assuming $w = X^T\alpha$ ($X^T\alpha$)

$$\text{eq } (X^T X + \lambda I_D)\boxed{w} \overset{?}{=} X^T y$$

$$\left[X^T X + \lambda I_D\right] \cdot X^T \alpha = X^T X X^T \alpha + \lambda X^T \alpha$$

$$= X^T\left(X X^T \alpha + \lambda I_n \alpha\right) = X^T\left(X X^T + \lambda I_N\right)\boxed{\alpha}$$

($\alpha = \left[X X^T + \lambda I_N\right]^{-1} y$)

$$= X^T \underbrace{\left[X X^T + \lambda I_n\right]}_{\text{same}} \underbrace{\left[X X^T + \lambda I_N\right]^{-1}}_{\text{same}} \cdot y$$

$$= X^T y \quad \checkmark$$

dual L_2-regression (Kernelized Ridge Regression, HW5)

α_i = dual var $\quad w = X^T \alpha$

$i = 1:N$

$$h(x) = x \cdot w = x \cdot \boxed{X^T \alpha} = \boxed{x \cdot X^T} \alpha = K\alpha$$

(w = $X^T\alpha$; primal)

dual

$$h(z) = \sum_{i=1}^{N} \alpha_i K(x_i, z)$$

= function (similarities) test for datapoint z

- train

$$\alpha = [XX^T + \lambda I_n]^{-1} y$$

use K = kernel valid other than XX^T

train $\alpha = [K + \lambda I_n]^{-1} y$ test $h(z) = \sum \alpha_i K(z, x_i)$

K = Gaussian, or K = polynomial

Recall Perceptron update rule (May 28 Lecture)

Perceptron problem (modified data)
- Given N vectors $x_1, x_2, \dots x_N$
- find hyplane w s.t. $x_i w \geq 0$ $\forall i$
(puts all points on pos side)

$$\text{ERROR } J(w) = \sum_{x_i} -x_i w \geq 0$$

x_i mistake: $x_i w < 0$

pos: how far from hyplane x_i is

not mistake (correct) $x_i w \geq 0$

OBS: no error $\Rightarrow$ no mistakes
$\Rightarrow J(w) = 0$

update: Every mistake x_i ($x_i w < 0$) gets added to w = normal of hyplane

Gradient Desc update

$$\frac{\partial J}{\partial w^d} = -\frac{\partial\, x_i w}{\partial w^d} = -\frac{\partial \left[\sum_d x_i^d w^d\right]}{\partial w^d}$$

$$= -x_i^d \quad \text{if } x_i = \text{mistake}$$

$$\frac{\partial J}{\partial w} = -x_i^T$$

Vector

GD update for datap x_i

$$w_{new} = w_{old} + \lambda \cdot x_i^T$$

all datapoints x_i = mistakes

$$w_{new} = w_{old} + \lambda \sum_{\substack{x_i \text{ mistakes} \\ \{x_i w < 0\}}} x_i^T$$

Perceptron Algorithm

while (condition)

step k : x_k misclassified

$$w^{new} = w^{old} + x_k$$

mechanics: count α_i = # steps where x_i has been used for update.

$$w_{final} = \underbrace{0}_{w\text{-init}} + \sum_{i=1}^{N} \alpha_i \cdot x_i = X^T \cdot \alpha$$

α_i: how many times counts for each datapoint; x_i: add to w

$$= \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_N x_N$$

α_1: # times x_1 has been added to w

$$h(x) = x \cdot w = x \cdot (X^T \alpha) = k \cdot \alpha$$

$$(x X^T)\alpha \dots$$

use any valid kernel!