What do we need to learn from data?

- a pattern ← not magic
- not solvable analytically ← not magic.
- data

Lecture 2

Fundamentals of learning feasibility

Is learning feasible?

probabalistic! can we extrapolate?

bin — red & green marbles

$P[\text{red}] = \mu$

$P[\text{green}] = 1 - \mu$

μ is unknown

N picks independantly to gen. sample

fraction of red in sample is ν

does ν help with μ? yes obv but is sampling problem

possible vs probable ← almost certain is good enough

What does ν say about μ?

in large N, ν converges to μ within ϵ

FORMALLY: $P[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$

ϵ squared term kills N

HOEFFDING'S INEQUALITY

all extrapolation is driven by tolerance and sample size. "$\mu \approx \nu$" is probably approximately correct.

- valid for all N and ϵ
- bound independant of μ
- tradeoff between N and ϵ and the bound.
  smaller N, bigger ϵ
- $\nu \approx \mu \Rightarrow \mu \approx \nu$ (when you run exp. μ is unknown. μ effects ν, not vice versa, but symmetric so our exps work)

HOW DOES THIS CONNECT TO LEARNING?

unknown μ which is a function

$f: X \to Y$

each marble is point $x \in \mathcal{X}$

bin is our input space, so is $\mathcal{X}$

o: $h(x) = f(x)$ (HYPOTHESIS CORRECT)

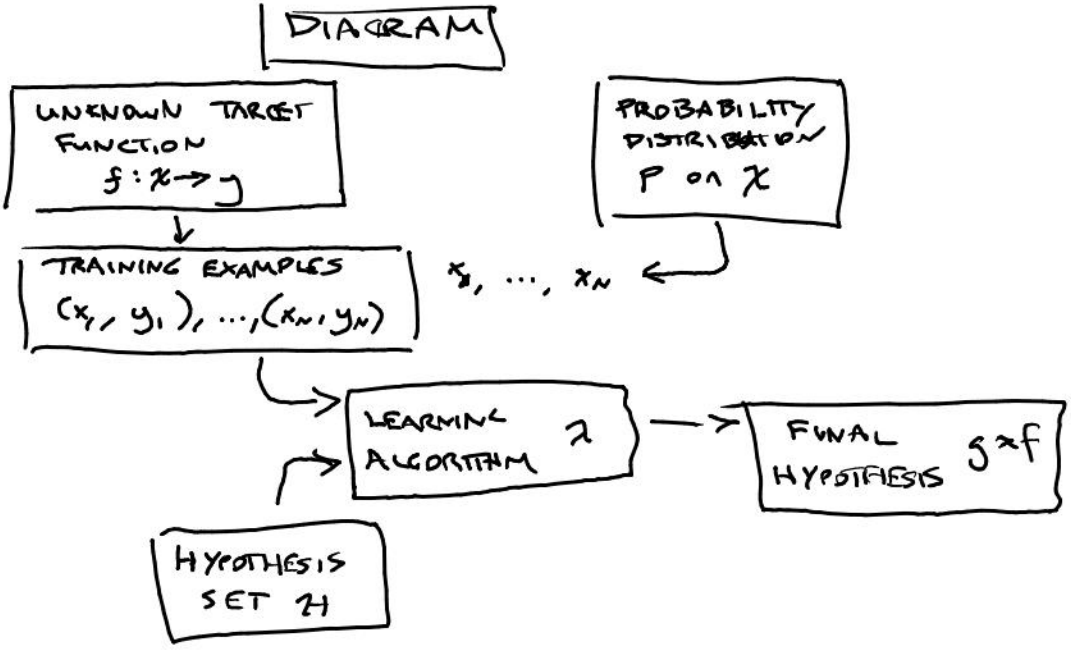
•: $h(x) \neq f(x)$

So we have a mapping for how accurate we are. introduces probability to learning

we are not restricting probability dists.

we don't have to know what it is.

hoeffding lets us do this because it's all sample & tolerance.



but we are not done!

h is fixed!

for <u>this</u> h, ν generalises to μ

THIS IS VERIFICATION, <u>NOT</u> LEARNING

DOES MY HYPOTHESIS WORK, NOT IS THIS OPTIMAL

WE NEED TO FIND HYPOTHESIS IN WHOLE SPACE

NO guarantee that ν is small and we're picking from $\mathcal{H}$

multiple bins? bin $\mathcal{H}_1, \ldots, \mathcal{H}_N$ generating different μ

try to drive $\mu \to 1$

is learning just picking the right bin?

seems silly for searches through infinite hypothesis space

ν (in sample) denoted by $E_{in}(h)$

μ (out of sample) denoted by $E_{out}(h)$

$$P\left[|E_{in}(h) - E_{out}(h)| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}$$

I SEE SOMETHING BAD COMING!

Hoeffding doesn't apply to multiple bins

bin number drives probability of sample w/in tolerance, as bins → ∞ we guarantee bad hypothesis is found!

if you toss 1 coin 10 times, prob of 10 heads is ~ 0.1%

if you toss 1000 coins 10x each, P(10 H) = 63%

I'm just quickly calcing...

$1-(1-0.5^{10})^{1000} \rightarrow 62.35\%$ OK

SOLUTION! g is final hypothesis

$$P\left[\left|E_{in}(g)-E_{out}(g)\right|>\epsilon\right] \leq P\left[\left|E_{in}(h_1)-E_{out}(h_1)\right|>\epsilon\right]$$

UNION BOUND

$$\ldots$$

$$\left|E_{in}(h_M)-E_{out}(h_M)\right|>\epsilon\Big]$$

$$\leq \sum_{m=1}^{M} P\left[\left|E_{in}(h_m)-E_{out}(h_m)\right|>\epsilon\right]$$

THIS SUCKS THO!

$$P\left[\left|E_{in}(g)-E_{out}(g)\right|>\epsilon\right] \leq \sum_{m=1}^{M} 2e^{-2\epsilon^2 N}$$

this is a generalisation problem! bigger M decouples in-sample from out

PROBLEM SET!

1) d

2) a

3) d (if black ball 2/3 chance it's in the 2-black bag)

4) $(1-0.55)^{10} = 3.405 \times 10^{-4}$ b

5) $1-(1-(1-0.55)^{10})^{1000} \Rightarrow 0.288$ c

6) $X = \{0,1\}^3$

$y_n = f(x_n)$ for $n = 1..5$

	x_1	x_2	x_3	y_n
	0	0	0	0
	0	0	1	1
D	0	1	0	1
	0	1	1	0
	1	0	0	1
!D	1	0	1	?
	1	1	0	?
	1	1	1	?

[c] ← but I don't feel great about it

WRONG. WHY?

7) b
8) c
9) b
10) b

LECTURE 3: THE LINEAR MODEL 1

g is final hypothesis

$$P\left[\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right] \leq M\, 2e^{-2\epsilon^2 N}$$

WORST CASE, UNION BOUND

FOR INDEPENDENT EVENTS ← OBV LIKE OF ATTACK

THIS IS THE PRINCIPLE THAT THROUGH LEARNING WE CAN GENERALISE, NOT THE FINAL RESULT

LET'S DO SOMETHING PRACTICAL BEFORE FINISHING THEORY

- INPUT REPRESENTATION
- LINEAR CLASSIFICATION
- LINEAR REGRESSION
- NON-LINEAR TRANSFORMATION

A REAL DATA SET: [DATA 16×16 IMAGE OF NUMBERS]

HUMAN ERROR ~ 2.5%

HOW DO WE REPRESENT INPUT?

16×16 => 256 pixels

raw input $x = (x_0 \dots x_{256})$

linear model: $(w_0 \dots w_{256})$ 256 dimensional space is too big!

INPUT REPRESENTATION IS SIMPLIFICATION

FEATURES: EXTRACT USEFUL INFORMATION

MAYBE INTENSITY AND SYMMETRY

$x = (x_0, x_1, x_2)$

WE LOSE INFORMATION, BUT PROBABLY USELESS INFORMATION, AND WE'VE DROPPED 254 DIMENSIONS

SO $x = (x_0, x_1, x_2)$ x_1 INTENSITY x_2 SYMMETRY

ACCEPT SOME ERROR!

PLA: EVALUATION OF E_{in} & E_{out} BY ITERATION

OUR DATA IS NOT LINEARLY SEPERABLE ← NEVER CONVERGES

CUTOFF AFTER 1K RUNS. FINAL HYPOTHESIS IS NOT GOOD. LET'S MODIFY

THE 'POCKET' ALGORITHM ← PICKS 'BEST' INTERMEDIATE HYPOTHESIS (E_{in} MIN).

MUCH BETTER

LINEAR REGRESSION

CLASSIFICATION: CREDIT APPROVAL (Y/N)

REGRESSION: CREDIT LINE (dollar amount)

INPUT x = age, salary, ..., debt

LINEAR REGRESSION OUTPUT: $h(x) = \sum_{i=0}^{d} w_i x_i = w^T x$

MAKES ALGORITHM LINEAR!

DATASET: HISTORICAL DECISIONS

$(x_1, y_1) \dots (x_n, y_n)$

$y_n \in \mathbb{R}$ is credit line for customer n

HOW TO MEASURE ERROR

how well does $h(x) = w^T x$ approximate $f(x)$?

use squared error in linear regression $(h(x) - f(x))^2$

(just standard rather than considered... for now)

in-sample error: $E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} h(x_n) - y_n)^2$

LINEAR REGRESSION TRIES TO PRODUCE A LINE THAT MINIMISES ERROR. HYPERPLANE of D-1

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} (w^T x_n - y_n)^2$$

$$= \frac{1}{N} ||Xw - y||^2$$

where $X = \begin{bmatrix} - x_1^T - \\ - x_2^T - \\ \cdots \\ - x_n^T - \end{bmatrix}$ $y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$

FEW PARAMETERS, MANY EXAMPLES

MINIMIZING

$$E_{in}(w) = \frac{1}{N} \| Xw - y \|^2$$

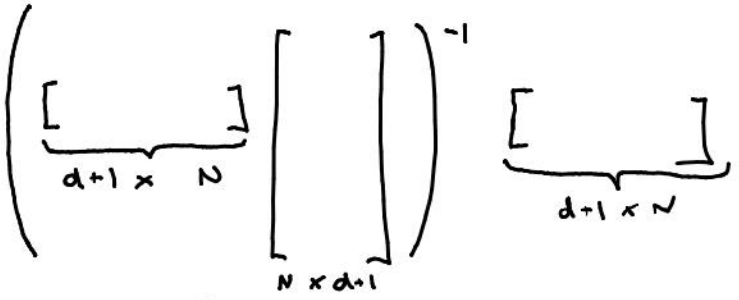
$$\nabla E_{in}(w) = \frac{2}{N} X^T (Xw - y) = 0 \leftarrow \text{VECTORS!}$$

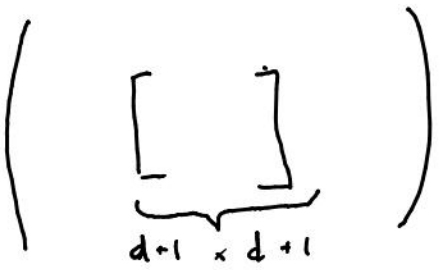
WHERE DOES THE TRANSPOSE COME FROM?

INVERTIBLE, PROBABY $\rightarrow X^T X w = X^T y$

$$w = X^+ y \quad \text{where} \quad X^+ = (X^T X)^{-1} X^T$$

↑ THE PSEUDO-INVERSE OF X





THE LINEAR REGRESSION ALGORITHM

1. CONSTRUCT THE MATRIX X AND VECTOR y FROM THE DATA SET $(x_1, y_1) \cdots (x_N, y_N)$

$$X = \begin{bmatrix} -x_1^T- \\ \cdots \\ -x_N^T- \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \cdots \\ y_N \end{bmatrix}$$

input data matrix — target vector

2. COMPUTE THE PSEUDO-INVERSE $X^+ = (X^T X)^{-1} X^T$
3. RETURN $w = X^+ y$

Why is X INVERTIBLE? WOULD HAVE TO BE SUPER UNLUCKY TO HAVE ALL YOUR DATA POINTS DEPENDENT ON EACH OTHER

LINEAR REGRESSION FOR CLASSIFICATION

linear regression learns a real-valued function

$$y = f(x) \in \mathbb{R}$$

binary functions are real-valued! $\pm 1 \in \mathbb{R}$

use linear regression to get w where $w^T x_n \approx y_n = \pm 1$

in this case, $\text{sign}(w^T x_n)$ is likely to agree w/ $y_n = \pm 1$

good init weights for classification

NON LINEAR TRANSFORMATION

LINEAR IS LIMITED

DATA: MESSY, NON-SEPERABLE. CAN WE FIND A TRICK?

CREDIT LINE

years in residence

nonlinear $[x_1 < 1]$ and $[x_1 > 5]$

can we do this with linear models

LINEAR IN WHAT?

linear regression implements

$$\sum_{i=0}^{d} w_i x_i$$

linear classification implements

$$\text{sign}\left(\sum_{i=0}^{d} w_i x_i\right)$$

linear in x_i yes but more importantly

linear in w!

data is transformable because weights given to non-linear features have linear dependency

transform from constant to other constant to make new constant separable

THERE IS A CATCH!!

WHERE X^T COMES FROM WHEN WE DIFFERENTIATE $\|Xw - y\|^2$

LET'S IGNORE CHAIN RULE FIRST & EXPAND

$$(Xw - y)^T (Xw - y)$$

$$\underset{①}{w^T X^T X w} - \underset{②}{w^T X^T y} - \underset{③}{y^T X w} - \underset{④}{y^T y}$$

d/dw of each term

① quadratic matrix differentiation

$d/dx (x^T A x) = 2Ax$ when A is symmetric.

$X^T X$ is <u>always</u> symmetric.

$\therefore d/dw (w^T X^T X w) = \underline{2X^T X w}$

② SCALAR TRANSPOSE TRICK! THE SCALAR PRODUCED BY $w^T a$ IS THE SAME AS $a^T w$. SO $w^T X^T y$ IS THE FORM $a^T w$, DERIVATIVE IS $X^T y$. ③ SAME TRICK

④ TRIVIAL

①..④ : $2X^T (Xw - y)$

TWO KEY THINGS:

1) MATRIX MULT IS NOT COMMUTATIVE

BUT WHEN RESULT IS SCALAR WE CAN TRANSPOSE AND FLIP

2) DIFFERETIATING A LINEAR SCALAR FUNCTION $a^T w$ WITH RESPECT TO w RETURNS THE COEFFICIENT VECTOR A — THE THING MULTIPLYING w, EXTRACTED & RETURNED IN VECTOR FORM. THE TRANSPOSE APPEARS IN DIFFERENTIATION WHEN BECAUSE EXTRACTING LEFT-SIDED X HERE REQUIRED TRANSPOSITION

$$d/dw\,(Xw) = X^T$$
$$d/dw\,(w^T X) = X$$

CHECK MATRIX SHAPES BEFORE PERFORMING OPERATIONS!!

ROW MULTIPICATION $[- - -]\begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \end{bmatrix}$ ✓

$\begin{bmatrix} - & - & - \\ - & - & - \end{bmatrix}[- - -]$ ✗

MATRIX MULTIPLICATION RULE

COLS OF FIRST MUST EQUAL ROWS OF SECOND

RESULT WILL BE ROWS OF FIRST x COLS OF SECOND

IF MULTIPLICATION IS INVALID THE ARE TRICKS —

- TRANSPOTION
- RESHAPING
- BROADCASTING
- PSEUDO - INVERSE

LECTURE 4

signal: $\sum_{i=0}^{d} w_i x_i = w^T x$

$w_{lin} = \underbrace{(X^T X)^{-1} X^T}_{X^\dagger} y$ (one-step learning)

transforms $\tilde{w}^T x$ linear in w, so x can be transformed → x is a stack of constants

$(x_1, x_2) \xrightarrow{\phi} (x_1^2, x_2^2)$ might help separate data

ERROR AND NOISE

NONLINEAR TRANSFORMATION

original: $x_n \in \mathcal{X}$ 

transformed: $z_n = \phi(x_n) \in \mathcal{Z}$

new data is linearly separable. then we apply ϕ^{-1} to $\hat{g}(z) = \text{sign}(\tilde{w}^T z)$

in $\mathcal{X}$-space: $g(x) = \text{sign}(\tilde{w}^T \phi(x))$

WHAT TRANSFORMS TO WHAT?

$x = (x_0 \dots x_d) \xrightarrow{\phi} \underbrace{z(z_0 \dots z_{\tilde{d}})}$

POTENTIALLY HARD TO GENERALISE

$x_1 \dots x_n \xrightarrow{\phi} z_1 \dots z_n$

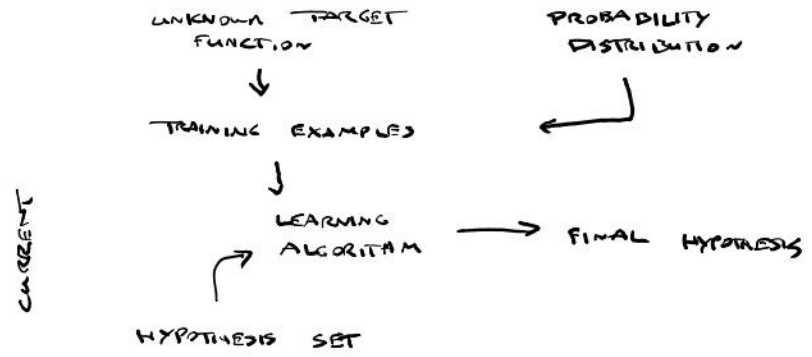
$y_1, \dots, y_N \xrightarrow{\phi} y_1 \dots y_N$ ← UNTOUCHED

NO WEIGHTS IN $\mathcal{X}$ $\quad \tilde{w} = (w_0 \dots w_{\tilde{d}})$

SO WHAT IS HYPOTHESIS?

$$g(x) = \text{sign}(\tilde{w}^T \phi(x))$$

ERROR MEASURES AND NOISY TARGETS



ERROR MEASURE

what does $h \approx f$ mean?

ERROR MEASURE: $E(h,f)$ ← we try to minimise

ALMOST ALWAYS POINTWISE DEFINITION: $e(h(x), f(x))$

EG: SQUARED ERROR: $e(h(x), f(x)) = (h(x) - f(x))^2$

BINARY " : $e(h(x), f(x)) = [\![h(x) \neq f(x)]\!]$

T/F

1 if true, 0 if false

POINTWISE TO OVERALL → OVERALL IS AVERAGE OF POINTWISE ERRORS

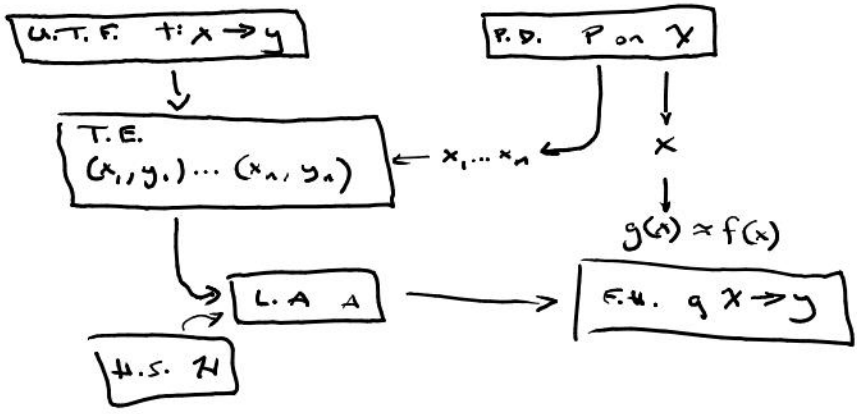
in-sample: $E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} e(h(x_n), f(x_n))$

↑ LOOKS LIKE SOMETHING WE'VE SEEN

OUT OF SAMPLE:

$$E_{out}(h) = \mathbb{E}_x\left[e(h(x), f(x))\right]$$

ADD POINTWISE ERROR TO LEARNING DIAGRAM.



ERROR COMES FROM PROBABILITY DISTRIBUTION THAT DRIVES TRAINING DATA. OR HOEFFDING DOESN'T WORK

CHOOSE ERROR MEASURE

FINGERPRINT VERIFICATION

$$\text{print} \to f \to \begin{cases} +1 & \text{You} \\ -1 \end{cases}$$

error types: false positive, false negative

how to penalise?

h \ f	$+1$	-1
$+1$	no error	false pos
-1	false neg	no error

← no inherit merit in picking one error function over another

IN A SUPERMARKET

COST OF FALSE POSITIVE LOW

COST OF FALSE NEGATIVE HIGH

h \ f	+1	-1
+1	0	1
-1	10	0

IN THE CIA

COST OF FALSE POSITIVE HIGH

COST OF FALSE NEGATIVE LOWER

h \ f	+1	-1
+1	0	1000
-1	1	0

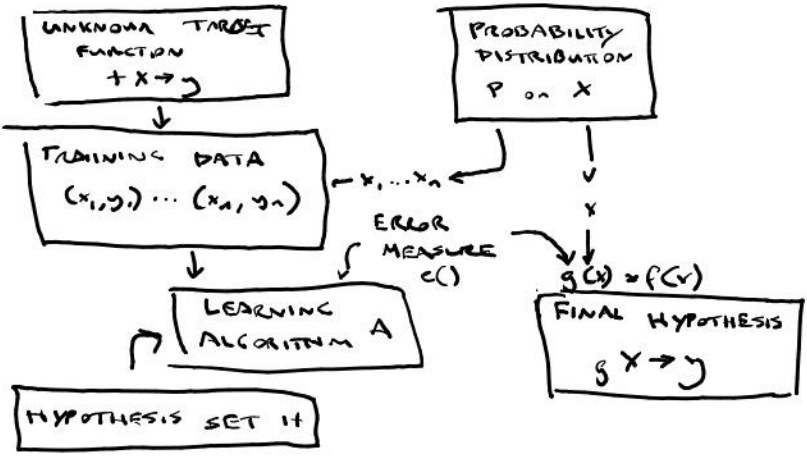
PRACTICAL LEARNING PROBLEM → ERROR MEASURE SPECIFIED BY USER

NOT ALWAYS POSSIBLE. WHAT ARE ALTERNATIVES?

PLAUSIBLE MEASURES: squared error ≡ (gaussian noise)

FRIENDLY MEASURES: closed-form solution, convex optimisation

MODIFY LEARNING DIAGRAM AGAIN



NOISY TARGETS

THE TARGET FUNCTION IS NOT ALWAYS A FUNCTION

(OBVIOUSLY!)

CREDIT APPROVAL EXAMPLE → TWO IDENTICAL CUSTOMERS IN TRAINING SET HAVE DIVERGENT OUTCOMES

INSTEAD OF $y = f(x)$ WE USE TARGET DISTRIBUTION

$$P(y|x)$$

(x, y) IS NOW GENERATED BY THE JOINT DISTRIBUTION

$$P(x)\,P(y|x)$$

noisy target = deterministic target $f(x) = \mathbb{E}(y|x)$ plus noise $y - f(x)$

deterministic function is a special case of noisy target $P(y|x) = 0$ unless $y = f(x)$

FINAL LEARNING DIAGRAM!

(for supervised learning)

DISTINCTION BETWEEN $P(y|x)$ AND $P(x)$

BOTH TARGET DISTRIBUTION AND $P(x)$ ARE UNKNOWN

CONVEYING PROBABILISTIC ASPECTS OF X AND Y

TARGET DISTRIBUTION $P(y|x)$ IS WHAT WE'RE TRYING TO LEARN.

THE INPUT DISTRIBUTION $P(x)$ QUANTIFIES RELATIVE IMPORTANCE OF POINT X. WE'RE NOT TRYING TO LEARN IT

MERGING $P(x)P(y|x)$ AS $P(x,y)$ MERGES CONCEPTS.

DANGEROUS!

PREAMBLE TO THEORY OF LEARNING

- learning is feasible. it is likely that

$$E_{out}(g) \approx E_{in}(g)$$

is this learning?

we need $g \approx f$, which means $E_{out}(g) \approx 0$

so what is $E_{out}(g) \approx E_{in}(g)$? generalisation

$E_{in}(g)$ is a proxy

THE 2 QUESTIONS

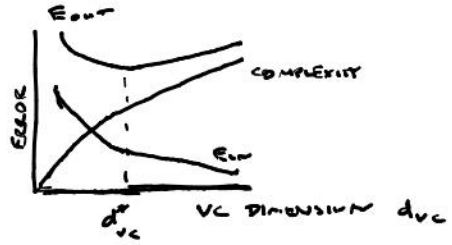
$E_{out}(g) \approx$ ACHIEVED THROUGH

$E_{out}(g) \approx E_{in}(g)$ and $E_{in}(g) \approx 0$

LEARNING IS THEREFORE TWO QUESTIONS.

1. CAN WE MAKE SURE $E_{out}(g)$ IS CLOSE TO $E_{in}(g)$
2. CAN WE MAKE $E_{in}(g)$ SMALL ENOUGH? (LOADED TERM!)

WHAT THE THEORY WILL ACHIEVE



characterise the feasibility of learning for infinite M

characterise the tradeoff ← REGULARISATION

model complexity	↑	E_{in} ↓
"	↑	$E_{out} - E_{in}$ ↑

HOMEWORK #2

1. b
2. d
3. OK WHEN $1-\mu$ ON λ
   " μ ON $1-\lambda$
   ERROR WHEN μ ON λ
   $1-\mu$ ON $1-\lambda$

   e
4. b ← if your function is binary and you have a 0.5 error you're just guessing!
5. c
6. C
7. a ← obvious but I should build
8. d
9. -1, 0, 0, 0.05, 1.6, 1.5 ← a
10. b

LECTURE 5: TRAINING AND TESTING

ERROR MEASURES SHOULD BE USER SPECIFIED

BUT WE CAN ALSO LOOK ANALYTICALLY/PRACTICALLY.

THEN WE PLUG INTO IN-SAMPLE AND OOS, E IS EXPECTED VALUE ACROSS (KNOWN) PROBABILITY DIST.

REVISION

NOISE FACTOR → $P(y|x)$ makes target distribution

$E_{out}(h) = E_{x,y}[e(h(x), y)]$; $(x_1, y_1) \dots (x_N, y_N)$ generated by $P(x, y) = P(y|x) \cdot P(x)$

THIS LECTURE

- FROM TRAINING TO TESTING
- KEY NOTION: BREAK POINT

TRAINING TO TESTING

- REVISION! SAMPLE PROBLEMS TO PREPARE YOU FOR A TEST. WHY DON'T WE JUST PRACTICE ON THE FINAL? B/C FINAL IS JUST WAY OF GAUGING LEARNING, NOT THE GOAL
- WHAT DOES THIS LOOK LIKE FORMALLY?

TESTING: PLAIN HOEFFDING

$$\mathbb{P}\left[|E_{in}-E_{out}|>\epsilon\right] \leq 2\, e^{-2\epsilon^2 N}$$

TRAINING: M-modified

$$\mathbb{P}\left[|E_{in}-E_{out}|>\epsilon\right] \leq 2M e^{-2\epsilon^2 N}$$

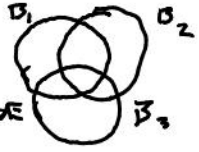
WE WOULD LIKE TO GET RID OF M BECAUSE M IS ALWAYS INFINITE!

WHERE DOES M COME FROM?

BAD EVENTS B_m ARE WHEN E_{in} DOES NOT TRACK E_{out}: "$|E_{in}(h_m) - E_{out}(h_m)| > \epsilon$"

THEN UNION BOUND APPLIED. BUT OBV BAD EVENTS ARE NOT INDEPENDANT!

i.e. $\mathbb{P}[B_1 \text{ or } B_2 \text{ or } \dots B_M]$



UNION BOUND IS CONSERVATIVE

$\sum_{n=1}^{M} B_n$ ← ASSUMES NO OVERLAP

NON-CONSERVATIVE BOUND NEEDS TO CHARACTERISE THE OVERLAP

CAN WE IMPROVE ON M?

YES! BAD EVENTS VERY OVERLAPPING.

PERCEPTRON HAS INFINITE M BUT THEY'RE TIGHTLY CORRELATED

FOR SMALL LINE CHANGE...

ΔE_{out}: change in area of error regions
ΔE_{in}: change in labels of data points
} BOTH TINY AND CORRELATED

$$|E_{in}(h_1) - E_{out}(h_1)| \approx |E_{in}(h_{min}) - E_{out}(h_{min})|$$

OVERLAPPING!

M REPLACEMENT TIME!

COMPLEXITY! INSTEAD OF WHOLE INPUT SPACE, RESTRICT ATTENTION TO THE SAMPLE. CONSIDER FINITE INPUT, AND COUNT NUMBER OF h. WE CALL THESE DICHOTOMIES

DICHOTOMIES: mini-h

$h : \mathcal{X} \to \{-1, +1\}$ ← hypothesis

$h : \{x_1, \dots x_n\} \to \{-1, +1\}$ ← dichotomies

$|\mathcal{H}|$ can be infinite for hypotheses

number of dichotomies $|\mathcal{H}(x_1, x_2 \dots x_N)| \to 2^N$

$$= 2 \cdot 2^N e^{-2\epsilon^2 N}$$ ← HOEFFDING

THE GROWTH FUNCTION

THE GROWTH FUCTION COUNTS MAX DICHOTOMIES FOR ANY HYPOTHESIS SET GIVEN N POINTS

$$m_{\mathcal{H}}(N) = \max_{x_1, \dots, x_N \in \mathcal{X}} \underbrace{|\mathcal{H}(x_1, \dots, x_N)|}_{\substack{\text{SET OF} \\ \text{DICHOTOMIES} \\ \text{(COUNT)}}}$$

SATISFIES

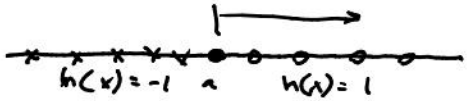
$m_{\mathcal{H}}(N) \leq 2^N$

$m_{\mathcal{H}}(3) = 8$

$m_{\mathcal{H}}(4) = 14$ ← WE CAN'T REACH SOME CONFIGURATIONS

BREAK POINT: SET OF ALL POSSIBLE HYPOTHESES STRONG THAN $\mathcal{H}$ PERCEPTRON

POSITIVE RAYS



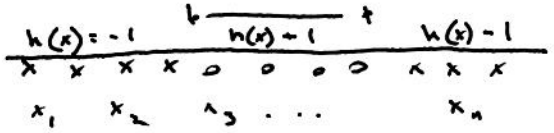
$x_1 \quad x_2 \quad x_3 \quad \dots \quad x_n$

$\mathcal{H}$ is set of $h = \mathbb{R} \to \{-1, +1\}$

$h(x) = \text{sign}(x - a)$

$m_{\mathcal{H}}(N) = N + 1$

POSITIVE INTERVALS



$\mathcal{H}$ is set $h: \mathbb{R} \to \{-1, -1\}$

$m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1 = \frac{1}{2}N^2 + \frac{1}{2}N + 1$

CONVEX SETS

$\mathcal{H}$ is set of $h: R^2 \to \{-1, -1\}$

$h(x) = +1$ is convex

$m_{\mathcal{H}}(N) = ?$

COLLAPSE INTO CIRCLE SO GROWTH FUNCTION IS

2^N

WEAKNESS OF GROWTH FUNCTION AS MAX IS

THAT BOUND IS NOT AS TIGHT AS POSSIBLE GIVEN

TRAINING DATA.

"SHATTERING" IS WHEN YOU GET ALL POSSIBLE

DICHOTOMIES

GROWTH FUNCTIONS

POSITIVE RAYS: $N+1$

INTERVALS: $\frac{1}{2}N^2 + \frac{1}{2}N + 1$

CONVEX SET: 2^N

MORE SOPHISTICATED → BIGGER GROWTH FUNCTION

WHAT HAPPENS WHEN YOU REPLACE M WITH $m_{\mathcal{H}}(n)$?

$$\mathbb{P}\left[|E_{in} - E_{out}| > \epsilon\right] \leq 2M e^{-2\epsilon^2 N}$$

$$\text{vs} \quad 2\, m_{\mathcal{H}}(n)\, e^{-2\epsilon^2 N}$$

if $\underbrace{m_{\mathcal{H}}(N) \text{ polynomial}}_{\text{WE NEED THIS!}}$ the exponential wins

BREAK POINT!

THE POINT AT WHICH AN $\mathcal{H}$ SET CAN'T REACH ALL POSSIBLE HYPOTHESES.

IF NO DATASET OF SIZE k CAN BE SHATTERED BY $\mathcal{H}$, THEN k IS A BREAK POINT FOR $\mathcal{H}$

$$m_{\mathcal{H}}(k) < 2^k$$

∴ PERCEPTRON: $k = 4$

BREAK POINT DRIVES LEARNING BEHAVIOR!

POSITIVE RAYS $k=2$

INTERVALS $k = 3$

CONVEX SET $k = \infty$

MAIN RESULT

NO BREAK POINT $\Rightarrow m_{\mathcal{H}}(N) = 2^N$

ANY " $\Rightarrow m_{\mathcal{H}}(N)$ IS polynomial in N (HASN'T TOLD US WHY YET)

SO ANY HYPOTHESIS SET WITH A BREAK POINT IS GENERALISABLE!

LECTURE 6: THE THEORY OF GENERALIZATION

· DICHOTOMIES: RESTRICT ATTENTION TO HYPOTHESES THAT IMPACT DATASET (FINITE)

GROWTH FUNCTION: $m_{\mathcal{H}}(N) = \max_{x_1, \dots x_N \in X} |\mathcal{H}(x_1, \dots, x_N)|$

most possible dichotomies

BREAK POINT: DATA POINTS THAT CANNOT BE SHATTERED BY HYPOTHESIS.

CHARACTERISES GENERALISATION

(1) · PROOF THE GROWTH FUNCTION $m_{\mathcal{H}}(N)$ IS POLYNOMIAL

(2) · PROOF THAT $m_{\mathcal{H}}(N)$ CAN REPLACE M

(1) TO SHOW: $m_{\mathcal{H}}(N)$ IS polynomial

WE SHOW: $m_{\mathcal{H}}(N) \leq \dots \leq \dots \leq$ a polynomial

KEY QUANTITY: $B(N,k)$: MAXIMUM NUMBER OF DICHOTOMIES ON N POINTS WITH BREAK POINT k

$B(N,k)$ IS PURELY COMBINATORIAL

RECURSIVE BOUND

ISOLATE LAST POINT

CONSIDER THE TABLE

	# ROWS	x_1	x_2	...	x_{N-1}	x_N
		+1	+1	...	+1	+1
S_1	α	-1	+1	...	+1	-1
S_1: ROWS THAT APPEAR ONCE (FOR $x_1 \dots x_{N-1}$)						
		+1	-1	...	+1	+1
		-1	-1	...	+1	+1
S_2^+	β			⋮		⋮
S_2:		+1	-1	...	+1	-1
		-1	-1	...	+1	-1
S_2^-	β			⋮		⋮

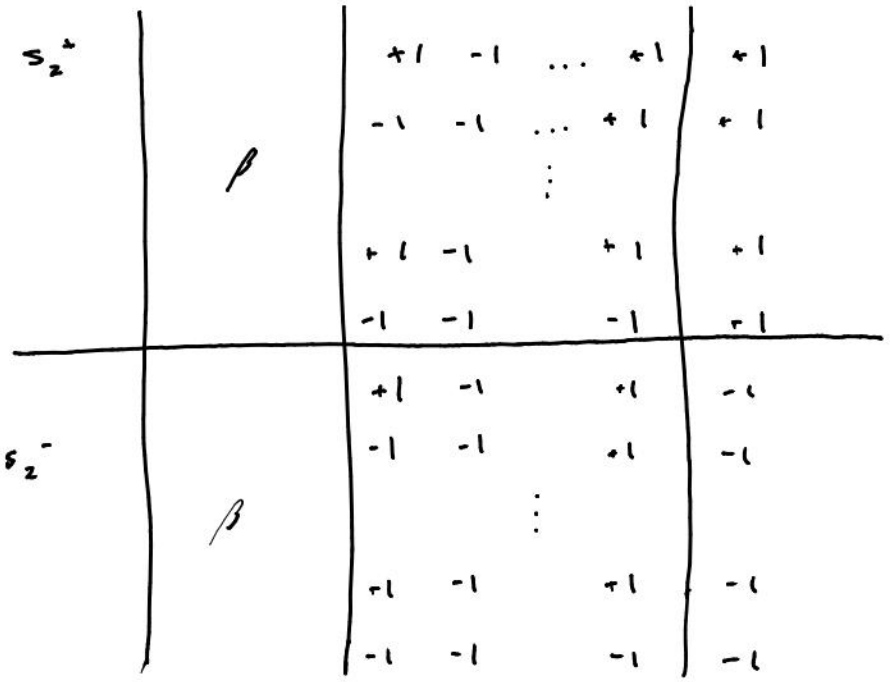
$$B(N,k) = \alpha + 2\beta$$

ESTIMATE α AND β

IF WE FOCUS ON COLS $x_1 \to x_{N-1}$, α REMAINS

BUT WE HAVE β TWICE. $\alpha + \beta \le B(N-1,k)$

NOW FOCUS ON $S_2 = S_2^+ \cup S_2^-$ ROWS



LOOK AT S_N^+ → MATRIX HAS BP OF K-1

IF YOU HAVE ALL POSSIBLE PATTERNS ON K-1 COLUMNS, ADDING X_n COL TAKES YOU TO K

$\beta \leq B(N-1, k-1)$

PUT IT TOGETHER AND DO IT AGAIN

$B(N,k) = \alpha + 2\beta \qquad \alpha + \beta \leq B(N-1, k)$

$\beta \leq B(N-1, k-1)$

$\therefore \; B(N,k) = B(N-1,k) + B(N-1, k-1)$

SOLUTION IS UPPER BOUND FOR ANY GROWTH FUNCTION WITH BREAK POINT

$$B(N,k) = B(N-1,k) + B(N-1,k-1)$$

N \ k	1	2	3	4	5	6	...
1	1	2	2	2	2	2	
2	1	3	4	4	4	4	
3	1	4	7				2^N
4	1	5	11	11			
5	1	6	16	22	.	.	
6	1	7	24	38			
⋮							

PUZZLE FROM LAST LECTURE

ANALYTIC SOLUTION:

THEOREM: $B(N,k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$

BOUNDARY CONDITIONS: EASY

INDUCTION STEP: $N-1$ $k-1$ k

$$\sum_{i=0}^{k-1} \binom{N}{i} = \sum_{i=0}^{k-1} \binom{N-1}{i} + \sum_{i=0}^{k-2} \binom{N-1}{i} \; ?$$

$$= 1 + \sum_{i=1}^{k-1} \binom{N-1}{i} + \sum_{i=1}^{k-1} \binom{N-1}{i-1}$$

$$= 1 + \sum_{i=1}^{k-1} \left[\binom{N-1}{i} + \binom{N-1}{i-1} \right]$$

REDUCE WITH COMBINATORIAL ARGUMENT

CHOOSE 10 PEOPLE FROM N IN A ROOM

N CHOOSE 10

OR CHOOSE 10 PEOPLE FROM N-1

N-1 CHOOSE 10

PLUS CHOOSE 9 PEOPLE FROM N-1

SO

$$1 + \sum_{i=1}^{k-1} \binom{N}{i} = \sum_{i=0}^{k-1} \binom{N}{i}$$

← POLYNOMIAL!

GIVEN

FOR $\mathcal{H}$, BREAK POINT IS FIXED

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

MAX POWER IS N^{k-1}

k IS CONSTANT;

THIS IS POLYNOMIAL IN N

THREE EXAMPLES $\sum_{i=0}^{k-1} \binom{N}{i}$

POSITIVE RAYS: $k = 2$

$$m_{\mathcal{H}}(N) = N+1 \leq \binom{N}{0} + \binom{N}{1} = 1 + N$$

POSITIVE INTERVAL: $k = 3$

$$m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1 < \frac{1}{2}N^2 + \frac{1}{2}N + 1$$

2D PERCEPTRON: $k = 4$

$$m_{\mathcal{H}}(N) = ? \quad \leq \frac{1}{6}N^3 + \frac{5}{6}N + 1$$

YOU SHOULD DO THESE REDUCTIONS

PROOF $m_{\mathcal{H}}(N)$ CAN REPLACE M

INSTEAD OF

$$\mathbb{P}\left[|E_{in} - E_{out}| > \epsilon\right] \leq 2M e^{-2\epsilon^2 N}$$

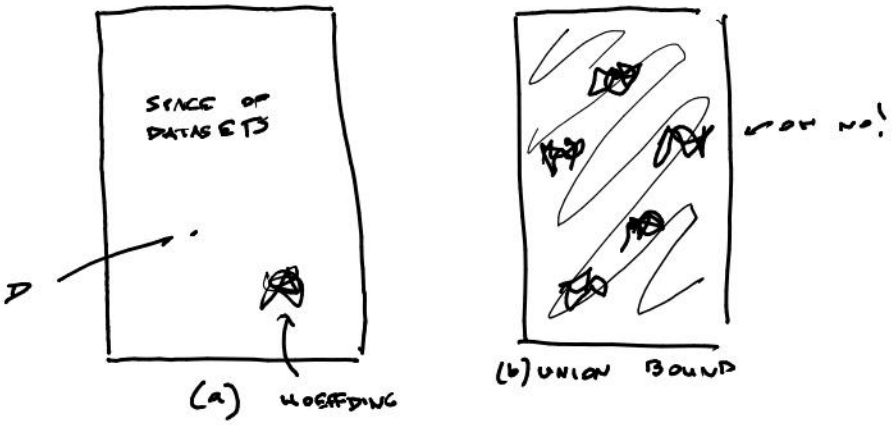
WE WANT

$$\ldots \leq 2 m_{\mathcal{H}}(N) e^{-2\epsilon^2 N}$$

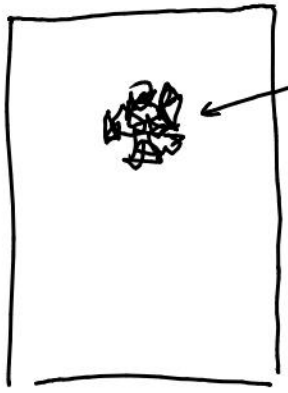
FORMAL PROOF IS A LOT!

- HOW DOES $m_{\mathcal{H}}(N)$ relate to overlaps?
- WHAT TO DO ABOUT E_{out}?
- PUT IT TOGETHER

PICTURES!



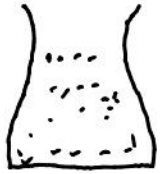
VC BOUND (c)



IF ANY POINT IS PAINTED, IT WILL BE PAINTED 100 TIMES. $A' \leq A/100$

WE COLLAPSE HYPOTHESES INTO DICHOTOMIES, EXCEPT FOR IT IS A THING.

$E_{out}(h)$



$E_{in}(h)$ ..... so $E_{in}(h)$ tracks $E'_{in}(h)$

$E'_{in}(h)$ ..... because both track E_{out}!

$E''_{in}(h)$ .....

it's the same problem! but in E_{in} only so it's back to dichotomies!

WE DON'T HAVE

$$\mathbb{P}\left[|E_{in}(h) - E_{out}(h)| > \epsilon\right] \leq 2 m_H(n) e^{-2\epsilon^2 N}$$

BUT RATHER:

$$\leq 4 m_H(2N) e^{-\frac{1}{8}\epsilon^2 N}$$

← BECAUSE OF THE $E_{in} \to E_{out}$

VAPNIK - CHERNOVENKIS INEQUALITY

WHY $\frac{1}{8}$? ϵ ENDS UP AS $\epsilon/4$ IN PROOF

IT'S UGLY BUT SHOWS LEARNING IS FEASIBLE

PROBLEM SET #3

1. $\mathbb{P}\left[|E_{in}(g) - E_{out}(g)| > \epsilon\right] \leq 2M e^{-2\epsilon^2 N}$

set $\epsilon = 0.05$, $M = 1$

what is N for 0.03

$0.03 \geq 2e^{-2(0.05^2)N}$

$0.03 \geq 2e^{-0.005N}$

$0.015 \geq e^{-0.005N}$

$\ln(0.015) \geq -0.05N$

$-4.2 \geq -0.005N$

$\therefore N \leq \boxed{840}$ b ✓

2. $0.03 \geq 20e^{-2(0.05)^2 N}$

$0.0015 \geq e^{-0.005N}$

$\therefore N \geq 1301$ c ✓

3. $0.00015 \geq e^{-0.005N}$

$\therefore N \geq 1761$ d ✓

4. Break point for a perceptron in 2D is 4. what about 3D? $\boxed{5}$

b ✓

5. Which of the following are possible formulae for a growth function $m_H(N)$:

i) $1+N$ iv) $2^{\lfloor N/2 \rfloor}$ NOT POLY

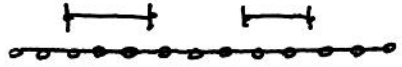
ii) $1+N+\binom{N}{2}$ v) 2^N

iii) $\sum_{i=0}^{\lfloor \sqrt{N} \rfloor} \binom{N}{i}$ NOT POLY

where $\lfloor u \rfloor$ is the biggest integer $\leq u$ and $\binom{m}{n} = 0$ when $m > M$

i, ii, v b ✓

6. $h: \mathbb{R} \to \{-1, +1\}$ and $h(x) = +1$ if point w/in intervals and -1 otherwise. smallest break point?



we know pos interval is 3; this gives us 2 more degrees of freedom. 5

c ✓

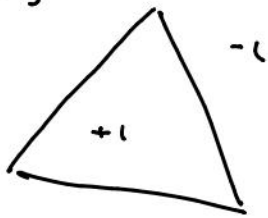
7. What is the growth function for the two-interval hypothesis set? place interval ends in 4 of N+1 spots

$\binom{N+1}{4} + \binom{N+1}{2} + 1$ c ✓

8. d ✓

9. triangle model:

$\mathbb{R}^2 \to \{-1, +1\}$



THE VERTICES SPLIT THE DATAPOINTS ON THE CIRCLE SELECTED FOR MAX DICHOMIES:

SO BREAK POINT IS $2 \cdot 3 + 1$

$\therefore 7$ ✓

10. COMPUTED THE GROWTH FUNCTION $m_H(N)$ for the learning model made up of two concentric circles in $\mathbb{R}^2$. SPECIFICALLY, $\mathcal{H}$ contains the functions which are $+1$ for $a^2 \leq x_1^2 + x_2^2 \leq b^2$ and -1 otherwise

is this different from positive intervals? I don't think so. $\therefore m_H(N) = 2^{N+1} + 1$

this wouldn't work w/ non-zero origins or ellipses but no info is encoded in angular direction

LECTURE 7: THE VC DIMENSION

RECAP: VC INEQUALITY:

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 4\, m_{\mathcal{H}}(2N)\, e^{-\frac{1}{8}\epsilon^2 N}$$

where $m_{\mathcal{H}}(N)$ is polynomial when k exists;

ie $m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$

YOU WILL REMEMBER THE VC DIMENSION!

- DEFINITION
- VC DIMENSION OF PERCEPTRONS
- INTERPRETATION
- GENERALISATION BOUNDS ← KEY

VC DIMENSION IS A NUMBER DEFINED FOR $\mathcal{H}$, DENOTED BY $d_{VC}(\mathcal{H})$; THE MOST POINTS $\mathcal{H}$ CAN SHATTER. THE LARGEST VALUE OF N FOR WHICH $m_{\mathcal{H}}(N) = 2^N$.

is this just k-1 or am I missing something?

$N \leq d_{VC}(\mathcal{H}) \Rightarrow \mathcal{H}$ can shatter N points

$N > d_{VC}(\mathcal{H}) \Rightarrow N$ is a break point of $\mathcal{H}$

THE GROWTH FUNCTION

in terms of k

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

in terms of d_{vc}

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{d_{vc}} \binom{N}{i}$$

d_{vc} is order of polynomial which bounds growth function

EXAMPLE: POSITIVE RAYS: $d_{vc} = 1$

2D PERCEPTRON: $d_{vc} = 3$

2D CONVEX SETS: $d_{vc} = \infty$ ← PESSIMISTIC UPPER BOUND

THE VC DIMENSION AND LEARNING

FINITE $d_{vc} \Rightarrow g \in \mathcal{H}$ will generalise
(FINAL HYPOTHESIS)

- INDEPENDENT OF THE LEARNING ALGORITHM
- " INPUT DISTRIBUTION

(BECAUSE WE ARE PICKING SAMPLES TO MAXIMISE DICHOTOMIES)

- " TARGET FUNCTION
- d_{vc}, N ↓, g ARE ALL THAT'S LEFT

↓
DRIVES VC INEQUALITY PROBABALISTICLY

d_{VC} FOR PERCEPTRONS

FOR $d=2$, $d_{VC}=3$

IN GENERAL, $d_{VC} = d+1$ ← LET'S SHOW WHY

$d_{VC} \leq d+1$ (1)

$d_{VC} \geq d+1$ (2)

(1) A SET OF $N = d+1$ POINTS IN $\mathbb{R}^d$ SHATTERED BY THE PERCEPTRON

$$X = \begin{bmatrix} -x_1^T- \\ -x_2^T- \\ \vdots \\ -x_{d+1}^T- \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ & & \vdots & & \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

CHOOSE SUCH THAT X IS INVERTIBLE

FOR ANY $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{d+1} \end{bmatrix} = \begin{bmatrix} \pm 1 \\ \pm 1 \\ \vdots \\ \pm 1 \end{bmatrix}$, can we find a vector w satisfying $\text{sign}(Xw) = y$

DROP THE SIGN! $Xw = y$ and X is invertible

$\therefore w = X^{-1}y$

SO WE CAN SHATTER THESE POINTS.

THIS IMPLIES WHAT?

$d_{VC} \geq d+1$ ① ✓

② SHOW THAT $d_{VC} \leq d+1$ BY SHOWING WE CANNOT SHATTER ANY SET OF $d+2$ POINTS

FOR ANY $d+2$ POINTS,

$$x_1, \cdots, x_{d+1}, x_{d+2}$$

MORE PANTS THAN DIMENSIONS ∴

$$x_j = \sum_{i \neq j} a_i x_i$$ ← AT LEAST ONE POINT IS LINEAR COMBO OF OTHERS

WHERE NOT ALL THE a_i's ARE ZERO

(we know because first component of perceptron is fixed 1)

CONSIDER DICHOTOMY:

x_i's WITH NON-ZERO a_i, get $y_i = \text{sign}(a_i)$

and x_j gets $y_j = -1$

NO PERCEPTRON CAN IMPLEMENT THIS

$$x_j = \sum_{i \neq j} a_i x_i \Rightarrow w^T x_j = \sum_{i \neq j} a_i w^T x$$

if $y_i = \text{sign}(w^T x_i) = \text{sign}(a_i)$, then $a_i w^T x_i > 0$

this forces $\sum_{i \neq j} a_i w^T x_i > 0$ ← UH OH!!

$y_j = \text{sign}(w^T x_j) = +1$ ← ∴ DICHOTOMY DOESN'T WORK, CAN'T SHATTER

① & ② $\Rightarrow$ $d_{vc} = d + 1$

THIS IS THE NUMBER OF PARAMETERS IN PERCEPTRON $w_0, w_1, \ldots, w_{d+1}$

INTERPETATION OF VC DIMENSION

① DEGREES OF FREEDOM:

PARAMETERS CREATE DEGREES OF FREEDOM

# of params: analog degrees of freedom

d_{vc}: equivalent 'binary' degrees of freedom

So ABSTRACTS MECHANICS OF MODEL INTO DICHOTOMIES

POSITIVE RAYS: ONE DEGREE OF FREEDOM ($d_{VC} = 1$)

" INTERVALS: TWO DOF ($d_{VC} = 2$)

NOT JUST PARAMETERS! PARAMETERS MAY NOT CONTRIBUTE TO DEGREES OF FREEDOM

STACKING S PERCEPTRONS GIVES YOU $S(D+1)$ PARAMETERS BUT DOESN'T ADD D.O.F

d_{VC} MEASURES EFFECTIVE NUMBER OF PARAMETERS

(2) NUMBER OF DATA POINTS NEEDED

VC INEQUALITY HAS TWO SMALL QUALITIES

$$P\left[|E_{in}(g) - E_{out}(g)| > \epsilon\right] \leq \underbrace{4 m_{\mathcal{H}}(2N) c^{-\frac{1}{8}\epsilon^2 N}}_{\delta \text{ — 2ND}}$$

IF WE WANT PARTICULAR ϵ & δ, HOW DOES N DEPEND ON d_{VC}?

SIMPLE & WHEN d_{VC} EXISTS

$N^{d_c - N}$

FIX $N^d e^{-N}$ = SMALL VALUE

HOW DOES N CHANGE WITH d?

BOUND IS ~ PROPORTIONAL TO $d!$

PRACTICAL OBSERVATION

THE QUANTITY WE'RE TRYING TO BOUND FOLLOWS THE BOUND ← WELL....

RULE OF THUMB: $N \geq 10\, d_{VC}$ GETS YOU INTO INTERESTING REGION

GENERALISATION THEOREM

START FROM VC INEQUALITY

$$\mathbb{P}\left[|E_{in}(g) - E_{out}(g)| > \epsilon\right] \leq \underbrace{4 m_H(2N) e^{-\frac{1}{8}\epsilon^2 N}}_{\delta}$$

GET ϵ IN TERMS OF δ

$$\delta = 4 m_H(2N) e^{-\frac{1}{8}\epsilon^2 N} \Rightarrow \underbrace{\sqrt{\frac{8}{N} \ln \frac{4 m_H(2N)}{\delta}}}_{\Omega} = \epsilon$$

WITH PROBABILITY $\geq 1-\delta$, $|E_{in} - E_{out}| \leq \Omega(N, \mathcal{H}, \delta)$

GENERALISATION BOUND

$E_{out} - E_{in} \leq \underbrace{\Omega(N, \mathcal{H}, \delta)}$ ← WE DROP ABSOLUTE VALUE BECAUSE $E_{out} \geq E_{in}$ FOR

WE KNOW BEHAVIOUR: WHAT WE CARE ABOUT CONDENSE

WITH PROBABILITY $\geq 1-\delta$

$$E_{out} \leq E_{in} + \Omega$$

WE DERIVE REGULARISATION FROM THIS!

(FOR ANOTHER TIME)

EXTRAS FROM LECTURE 7

SHATTERING - MATHEMATICAL DEFINITION

$\mathcal{H}$ SHATTERS A SET OF POINTS $\{x_1 \dots x_n\}$

IF AND ONLY IF FOR EVERY POSSIBLE LABELLING $y \in \{\pm 1\}^N$ THERE EXISTS SOME $h \in \mathcal{H}$ SUCH THAT $h(x_i)$ FOR ALL i

SO FOR PERCEPTRON WITH y (LABELS), FIND A w SUCH THAT $\underbrace{\text{SIGN}(Xw)}_{\text{CLASSIFIER}} = y$

$D_{VC} \leq d+1$ FOR PERCEPTRON

$x_s = \sum a_i x_i$ ← COMBO OF OTHER POINTS

$w^T x_s = \sum_i a_i (w^T x_i)$

$S^+ = \underbrace{i : a_i \geq 0}_{+1}$, $S^- = \underbrace{i : a_i < 0}_{-1}$

$x_s = -1$

DICHOTOMY: $w^T x > 0$ for $i \in S^+$

$w^T x < 0$ for $i \in S^-$

$w^T x_s < 0$

SO: $w^T x_s = \sum_{+} a_i (w^T x_i^+) + \sum_{+} a_i (w^T x_i^-) - \sum_{0} a_i (w^T x_i^0)$

$\therefore w^T x_s$ can't be less than zero, no shattering

LECTURE 8: BIAS-VARIANCE TRADEOFF

RECAP: WE HAVE GENERALISATION BOUND

$$E_{out} \leq E_{in} + \Omega(N, \epsilon, d_{vc})$$

$N \propto d_{vc}$ is interesting - what drives this?

B-V IS A STANDALONE THEORY W/ DIFFERENT APPROACH THAN VC.

- BIAS & VARIANCE (1)
- LEARNING CURVES (2)

(1) BIAS AND VARIANCE

WE HAVE BEEN THINKING ABOUT AN APPROXIMATION AND GENERALISATION TRADEOFF —

SMALL E_{out}: good approx. of f out of sample

MORE COMPLEX $\mathcal{H}$ => better chance to approx. f

LESS " => better chance of generalising

VC ONE APPROACH: $E_{out} \leq E_{in} + \Omega$

B-V IS ANOTHER: decompose E_{out} into

(1) How well $\mathcal{H}$ can approximate f

(2) How well we can hope to get h

B-V APPLIES TO REAL-VALUED FUNCTIONS, UNLIKE OUR WORK SO FAR WITH VC DIMENSION, AND USES SQUARED ERROR (REQUIRED FOR DECOMP)

START WITH E_{out}

MAKE DEPENDENCE ON DATASET EXPLICIT

DIFFERENCE BETWEEN HYPOTHESIS & TARGET

$$E_{out}(g^{(D)}) = \mathbb{E}_x\left[\left(g^{(D)}(x) - f(x)\right)^2\right]$$

$$\mathbb{E}_D\left[E_{out}(g^{(D)})\right] = \mathbb{E}_D\left[\mathbb{E}_x\left[\left(g^{(D)}(x) - f(x)\right)^2\right]\right]$$

we want dependence to go away!

$$= \underbrace{\mathbb{E}_x\left[\mathbb{E}_D\left[g^{(D)}(x) - f(x)^2\right]\right]}$$

CAN SWAP INTEGRATIONS

LET'S FOCUS ON $\mathbb{E}_D\left[g^{(D)}(x) - f(x)\right)^2]$. TO EVALUATE WE MUST CONSIDER "AVERAGE HYPOTHESIS" $\bar{g}(x) = \mathbb{E}_D\left[g^{(D)}(x)\right]$

WHAT g WOULD WE GET ON AVERAGE ON INFINITE DATASETS?

IMAGINE MANY DATASETS $D_1, D_2, \ldots D_K$ BUT YOU CAN ONLY LEARN ONE AT A TIME

$$\bar{g}(x) \approx \frac{1}{K}\sum_{1}^{K} g^{(D_K)}(x)$$

USING $\bar{g}(x)$

$$\mathbb{E}_D\left[\left(g^{(D)}(x) - f(x)\right)^2\right] = \mathbb{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x) + \bar{g}(x) + f(x)\right)^2\right]$$

$$= \mathbb{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2 + \left(\bar{g}(x) - f(x)\right)^2 + 2\underbrace{\left(g^{(D)}(x) - \bar{g}(x)\right)\left(\bar{g}(x) - f(x)\right)}_{\text{CROSS!}}\right]$$

TREAT $\bar{g}(x)$ & $f(x)$ AS CONSTANTS; $\mathbb{E}_D$ of CONSTANT IS ITSELF

BUT! WE CAN REMEMBER THAT $\mathbb{E}_D\, g^{(D)}(x) = \bar{g}(x)$ AND WE KNOW EXPECTED VALUE SUMS;

$2\left(\bar{g}(x) - \bar{g}(x)\right) = 0$, BYE BYE CROSS TERM!

VARIANCE TERM →

$$= \mathbb{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2\right] + \left(\bar{g}(x) - f(x)\right)^2$$

↑ BIAS TERM.

$$\mathbb{E}_D\left[\left(g^{(D)}(x) - f(x)\right)^2\right] = \text{var}(x) + \text{bias}(x)$$

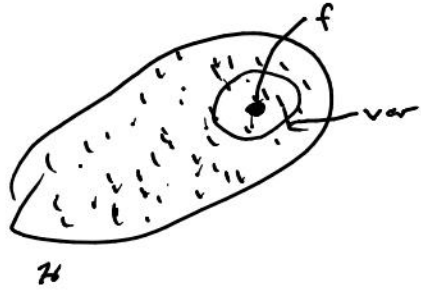
$$\therefore \quad \mathbb{E}_D\left[E_{out}\left(g^{(D)}\right)\right] = \mathbb{E}_x\left[\text{bias}(x) + \text{var}(x)\right]$$

$$= \text{bias} + \text{var}$$

THE TRADEOFF

$$\text{bias} = \mathbb{E}_x\left[\left(\bar{g}(x) - f(x)\right)^2\right]$$

$$\text{variance}: \mathbb{E}_x\left[\mathbb{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2\right]\right]$$



SMALL HYPOTHESIS TO BIGGER MEANS BIAS GOES DOWN & VARIANCE GOES UP

EXAMPLE: SINE TARGET

$f(x) = \sin(\pi x)$ $\quad f: [-1, 1] \to \mathbb{R}$

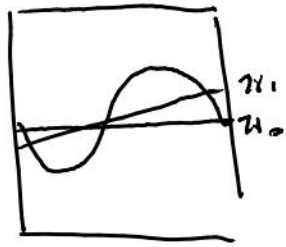
we get two training examples. $N = 2$

" hypothesis sets $\mathcal{H}_0: h(x) = b$

$\mathcal{H}_1: h(x) = ax + b$

which is better?

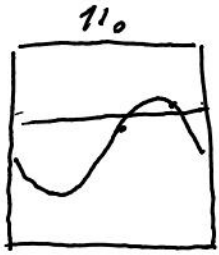
APPROXIMATION:



$\mathcal{H}_0$ $E_{out} = 0.5$

$\mathcal{H}_1$ $E_{out} = 0.2$

LEARNING:



ERROR DEPENDS ON DATASET!

BIAS & VARIANCE - $\mathcal{H}_0$

$\bar{g}(x)$ is the approximation in this case

big bias, small variance

$\mathcal{H}_1$: LINE CARNAGE!

$\mathcal{H}_0$: bias 0.5

var 0.25

0.75

$\mathcal{H}_1$: bias 0.21

var 1.69

1.90 ← uH oH!

MATCH THE MODEL COMPLEXITY TO THE DATA RESOURCES, NOT THE TARGET COMPLEXITY

② LEARNING CURVES

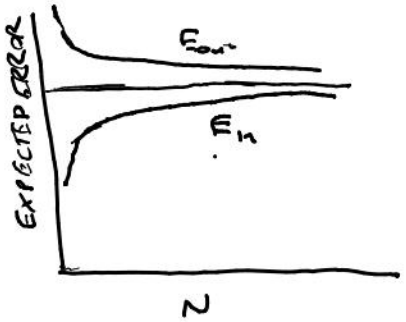
EXPECTED E_{out} AND E_{in}

DATASET $\mathcal{D}$ of size N

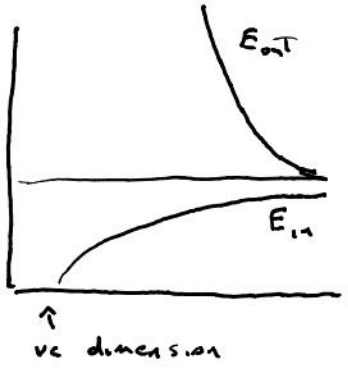
E_{out}: $\mathbb{E}_{\mathcal{D}}\left[E_{out}(g^{(\mathcal{D})})\right]$

E_{in}: $\mathbb{E}_{\mathcal{D}}\left[E_{in}(g^{(\mathcal{D})})\right]$

} how do they vary w/ N?

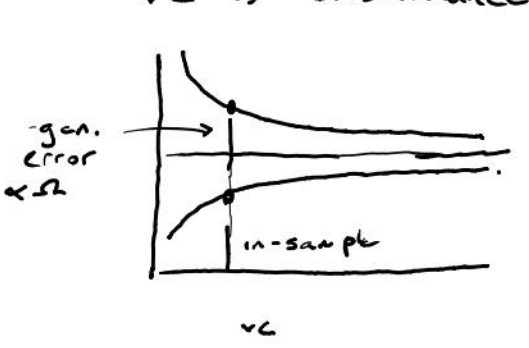


SIMPLE MODEL

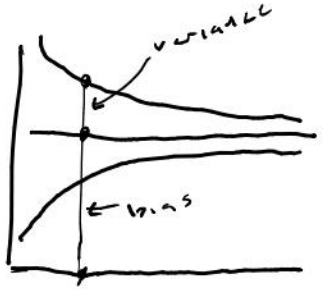


complex model

VC vs bias-variance



VC



b-v

LINEAR REGRESSION CASE

NOISY TARGET $y = w^{*T} x + \text{noise}$

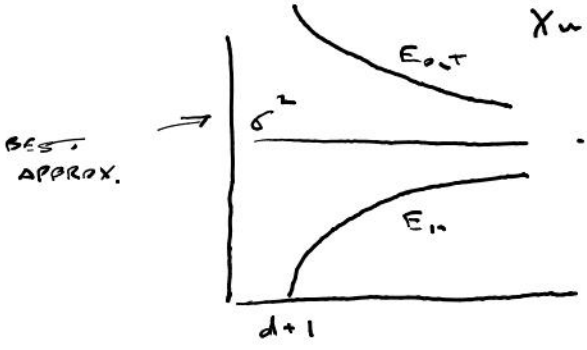
$D = \{(x_1, y_1), \dots, (x_N, y_N)\}$

$w = X^{\dagger} y$

in-sample error vector: $Xw - y$

"out of sample" error vector: generate fresh dataset w/ new noise y'

$Xw - y'$



exp. in-sample $= \sigma^2\left(1 - \frac{d+1}{N}\right)$

out-sample $= \sigma^2\left(1 + \frac{d+1}{N}\right)$

gen. error: $2\sigma^2\left(\frac{d+1}{N}\right)$

VC dimension!

PROBLEM SET #4

1: $d_{VC} = 10$, $\epsilon < 0.05$

$$0.05 \leq \sqrt{\frac{8}{N} \ln \frac{4 m_H(2N)}{0.95}}$$

$$0.0025N = 8 \ln\left(\frac{4 m_H(2N)}{0.95}\right) = 8 \ln\left(\frac{8N^{10}}{0.95}\right)$$

$$0.000125N = \ln(7.6 \cdot N^{10})$$

$$= \ln(7.6) + \ln(N^{10})$$

$$= \ln(7.6) + 10 \cdot \ln(N)$$

$$0.000125N = 2 + 10 \ln(N)$$

$$N = 6400 + 32{,}000 \ln(N)$$

$$N - 32{,}000 \ln(N) = 6400$$

420,000 (b)

2: d

3: c

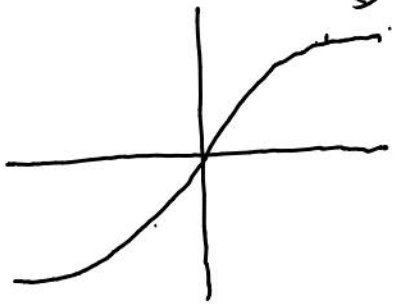
4: $f: [-1, 1] \to \mathbb{R}$ given by $f(x) = \sin(\pi x)$

getting something like $1.4x$ → (c)

5: -0.27 bias → (b)

6: ~ 0.2 variance → (a)

7:



ignore $ax-b$, b.

& ax^2

ax^2+b will have a decent bias but horrible variance

∴ (b)

8: Assume $q \geq 1$ is an integer and let $m_{\mathcal{H}}(1) = 2$

What is the hypothesis set whose growth function satisfies: $m_{\mathcal{H}}(N+1) = 2m_{\mathcal{H}}(N) - \binom{N}{q}$?

Recall that $\binom{M}{n} = 0$ when $n > M$

growth function starts polynomial regime at $N = q+1$; $d_{vc} = q$ (c)

9. $\mathcal{H}_{inter} = \mathcal{H}_1 \cap \mathcal{H}_2 \ldots \cap \mathcal{H}_N$. if $A \subseteq B \Rightarrow d_{vc}(A) \leq d_{vc}(B)$

∴ $\min\{d_{vc}(\mathcal{H}_k)\}_{k=1}^{4}$ is our best upper bound

∴ (b)

for union

10. lower bound is $\max \{d_{VC}(\mathcal{H}_k)\}_{k=1}^{K}$

K is a break point for set union;
not obvious contained within sum; ∴ K-1 is
in upper bound; ∴ e

LECTURE 9: THE LINEAR MODEL II

REMEMBER: N ∝ "VC DIMENSION"

WHERE WE ARE:

- LINEAR CLASSIFICATION ✓
- LINEAR REGRESSION ✓
- LOGISTIC REGRESSION
- NONLINEAR TRANSFORMS ~

CLEAN UP NET DIMENSIONALITY

$x = (x_0, x_1, \ldots, x_d) \xrightarrow{\Phi} z = (z_0, z_1, \ldots, z_{\tilde{d}})$

Each $z_i = \phi_i(x)$ ← any function; arbitrary length

$z = \Phi x$

EXAMPLE: $z = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$

final hypothesis $g(x)$ in $\mathcal{X}$ space:

$$\text{sign}\left(\tilde{w}^T \underline{\Phi}(x)\right) \quad \text{or} \quad \tilde{w}^T \underline{\Phi}(x)$$

GENERALISATION

$$x = (x_0, x_1, \dots, x_d) \xrightarrow{\underline{\Phi}} z = (z_0, z_1, \dots, z_{\tilde{d}})$$

$$\downarrow \qquad\qquad \downarrow$$

$$w \qquad\qquad \tilde{w}$$

$\tilde{w}$ much bigger than w

$d_{vc} = d + 1$ $\qquad$ $d_{vc} \leq \tilde{d} + 1$

TWO NON-SEPARABLE CASES

① 

② 

① use linear model in $\mathcal{X}$

accept $E_{in} > 0$ (α)

or

insist $E_{in} = 0$; go to high-dimensional $\mathcal{Z}$ (β)

β BAD IDEA FOR THIS DATA!

② $z = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$

6-dimensional; 2x examples required?

why not: $z = (1, x_1^2, x_2^2)$? ← $\tilde{d} = 3$

or: $z = (1, x_1^2 + x_2^2)$? ← $\tilde{d} = 2$

or: $z = (x_1^2 + x_2^2 - 0.6)$? ← $\tilde{d} = 1$???

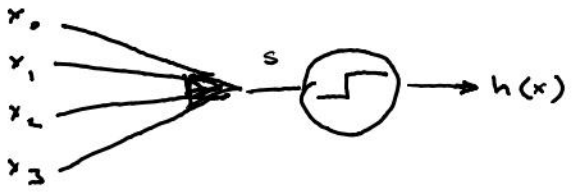
THIS DOESN'T WORK! YOU'RE CHERRY-PICKING HYPOTHESIS FROM $\mathcal{H}$! GENERALISATION STARTS WITH $\mathcal{H}$, NOT PRUNED DOWN; VC INEQUALITY STOPS APPLYING IF YOU SNOOP

SO HOW DO WE CHOOSE A MODEL?? (FOR LATER)

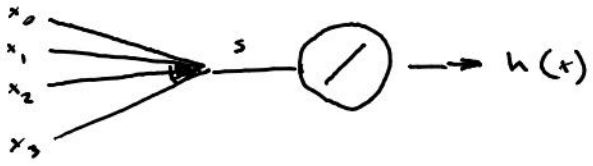
LOGISTIC REGRESSION

- THE MODEL (1)
- ERROR MEASURE (2)
- LEARNING ALGORITHM (3)

(1) $s = \sum_{i=0}^{d} w_i x_i$ linear classification $h(x) = \text{sign}(s)$

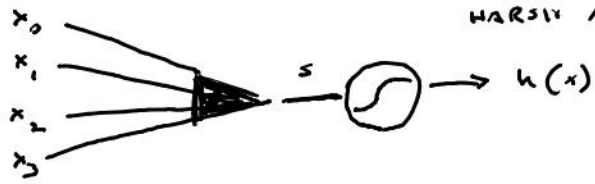


linear regression $h(x) = s$



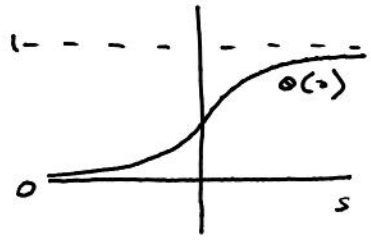
logistic regression $h(x) = \theta(s)$

↑ NON-LINEAR BUT NOT AS HARSH AS SIGN()



THE MAIN UTILITY IS THAT THE OUTPUT IS INTERPRETED AS A PROBABILITY

THE LOGISTIC FUNCTION θ



obv can be interpreted as a probability/uncertainty

$$\theta(s) = \frac{e^s}{1+e^s}$$

soft threshold
sigmoid

THIS FORMULA IS CHOSEN FOR FRIENDLY FUTURE MATHS

PROBABILITY INTERPRETATION

$h(x) = \theta(s)$ is interpreted as a probability

EXAMPLE: heart attacks

input: age, weight, etc

$\theta(s)$: probability of h.a. over some time

$s = w^T x$ "risk score"

THIS IS <u>GENUINE</u> PROBABILITY

DATA (x, y) WITH BINARY y, GENERATED BY NOISY TARGET

$$P(y|x) = \begin{cases} f(x) & \text{for } y = 1 \\ 1 - f(x) & \text{for } y = -1 \end{cases}$$

WE'RE TRYING TO LEARN f, WHICH MUST BE PROB!

$$f: \mathbb{R}^d \to [0,1]$$

LEARN $g(x) = \theta(w^T x) \approx f(x)$

NOW DO WE CHOOSE THE WEIGHTS?

ERROR MEASURE

for each (x, y), y generated by $f(x)$

plausible measure based on likelihood:

if $h = f$, how likely to get y from x?

this is never completely clean - live with it!

$$P(y|x) = \begin{cases} h(x) & \text{for } y = +1 \\ 1 - h(x) & \text{for } y = -1 \end{cases}$$

substitute $h(x) = \theta(w^T x)$, noting $\theta(-s) = 1 - \theta(s)$

use to simplify split: $P(x|y) = \theta(y w^T x)$

likelihood of $\mathcal{D} = (x_1, y_1), \ldots, (x_N, y_N)$ is

$$\prod_{n=1}^{N} P(y_n | x_n) = \prod_{n=1}^{N} \theta(y_n w^T x_n)$$

adjusting the weights to compromise between data points captures probability.

minimize $-\frac{1}{N} \ln \left(\prod_{n=1}^{N} \theta(y_n w^T x_n) \right)$

$$= \frac{1}{N} \sum_{n=1}^{N} \ln \left(\frac{1}{\theta(y_n w^T x_n)} \right) \qquad \left[\theta(s) = \frac{1}{1+e^{-s}} \right]$$

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\ln \left(1 + e^{-y_n w^T x_n} \right)}_{e(h(x_n), y_n)}$$

if sign agrees with risk factor, small error

$e(h(x_n), y_n)$ ↓ error measure

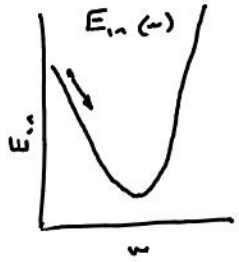
"cross-entropy" error

LOGISTIC REGRESSION ALGORITHM

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n w^T x_n}\right)$$

for linear regression we had a closed-form solution.

we don't have that here. let's use <u>gradient descent</u>



(generally applicable as long as surface is smooth)

start at $w(0)$; take step along steepest slope

you only have local information

fixed step size: small to avoid higher-order mess

$w(1) = w(0) + \eta \hat{v}$

what is the direction $\hat{v}$?

$$\Delta E_{in} = E_{in}(w(0) + \eta\hat{v}) - E_{in}(w(0))$$

$$= \eta \nabla E_{in}(w(0))^T \hat{v}$$ ← TAYLOR APPROX, IGNORE HIGH-ORDER

η - $\eta \| \nabla E_{in}(w(0))$ $\hat{v}$ is a unit vector

$$\hat{v} = -\frac{\nabla E_{in}(w(0))}{\| \nabla E_{in}(w(0)) \|}$$

STEP SIZE: TRADEOFF BETWEEN ACCURACY & SPEED

η too small: takes forever

η too large: may not converge, 2nd & 3rd order terms dominate

variable η - just right

η increasing with slope is best

instead of $\Delta w = \eta \hat{v}$

use $\Delta w = -\eta \nabla E_{in}(w(0))$

THIS IS A FIXED LEARNING RATE, NOT STEP SIZE

LOGISTIC REGRESSION ALGORITHM

1. INITIALISE WEIGHTS AT $t=0$ to $w(0)$
2. for $t=0, 1, 2, \ldots$ do
3. compute gradient $\nabla E_{in} = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n x_n}{1 + e^{y_n w^T(t) x_n}}$
4. update weights $w(t+1) = w(t) - \eta \nabla E_{in}$
5. iterate
6. return w

LECTURE 10: NEURAL NETWORKS

REVIEW: GRADIENT DESCENT

Q: IMPLEMENT GRADIENT?.

TERMINATION CRITERIA

NEURAL NETWORKS

- STOCHASTIC GRADIENT DESCENT
- NEURAL NETWORK MODEL
- BACKPROPAGATION

STOCHASTIC GRADIENT DESCENT

GD MINIMIZES $E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{e(h(x_n), y_n)}_{\ln(1+e^{-y_n w^T x_n}) \leftarrow \text{l.r.}}$

by iterative steps along $-\nabla E_{in}$

$\Delta w = -\eta \nabla E_{in}(w)$

∇E_{in} based on all examples

"batch" gr.

STOCHASTIC: PICK ONE AT A TIME.

APPLY GD to $e(h(x_n), y_n)$ ← LIKE PLA

"AVERAGE DIRECTION" $\mathbb{E}_n\left[-\nabla e(h(x_n), y_n)\right]$

PLUS NOISE → $= \frac{1}{N}\sum_{n=1}^{N} -\nabla e(h(x_n), y_n)$

$= -\nabla E_{in}$

HOPE NOISE CANCELS

MAIN BENEFIT: CHEAPER COMPUTATION

OTHER BENEFITS: RANDOMISATION — DON'T WANT TO BE TOO DETERMINISTIC, TRAPS IN LOCAL MINIMA / PLATEAUX

SIMPLE! RULE OF THUMB — $\eta = 0.1$ works

SGD IN ACTION

MOVIE RATINGS — user and movies have own profiles, reverse engineer ratings

user i, movie j, rating: r_{ij}

$u_{i1}, u_{i2} \dots u_{ik}$ $\quad$ $v_{j1}, v_{j2} \dots v_{jk}$

$$\left(r_{ij} - \sum_{k=1}^{k} u_{ik} v_{jk}\right)^2 = e_{ij}$$ ← minimising this

going rating by rating is stochastic gradient descent!

step η down gradient

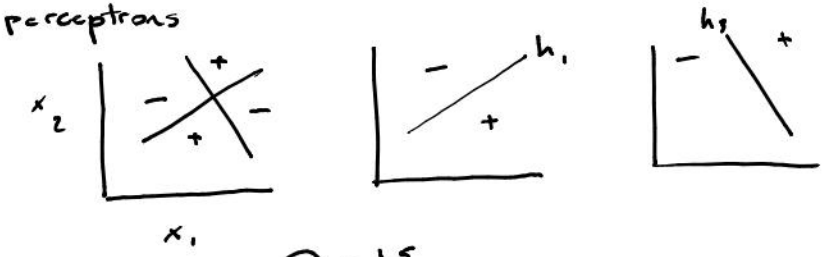
NEURAL NETWORK MODEL

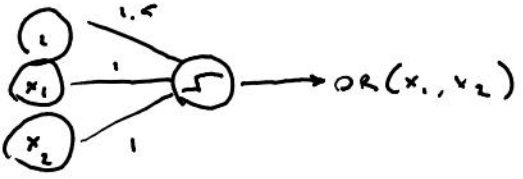
BIOLOGICAL INSPIRATION

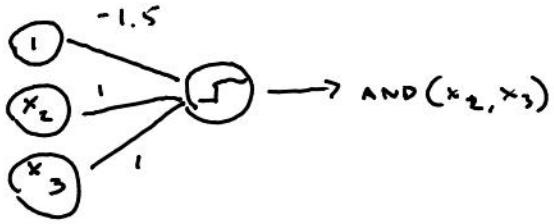
function → structure NEURONS AS PERCEPTRONS

but then pick what is relevant and take it :) don't get carried away with biology

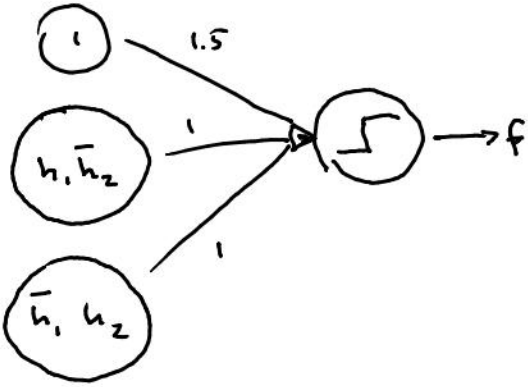
what can we do with combos of perceptrons

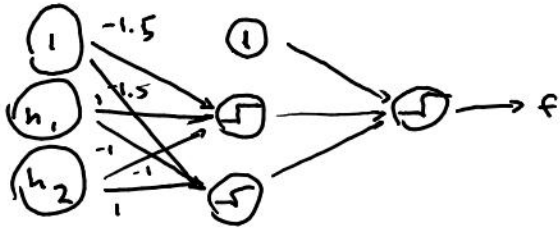






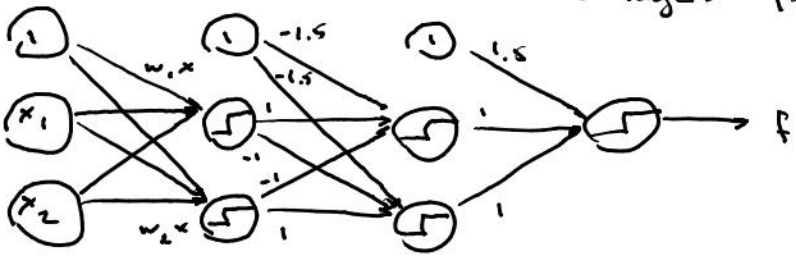
CREATING LAYERS

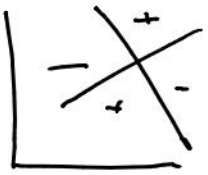




MULTILAYER PERCEPTRON

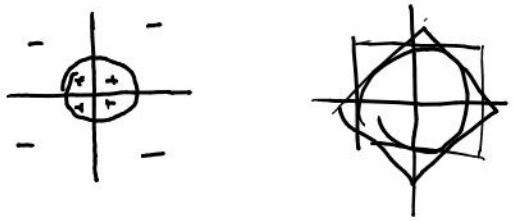
3 layers "feedforward"





solved!

POWERFUL!

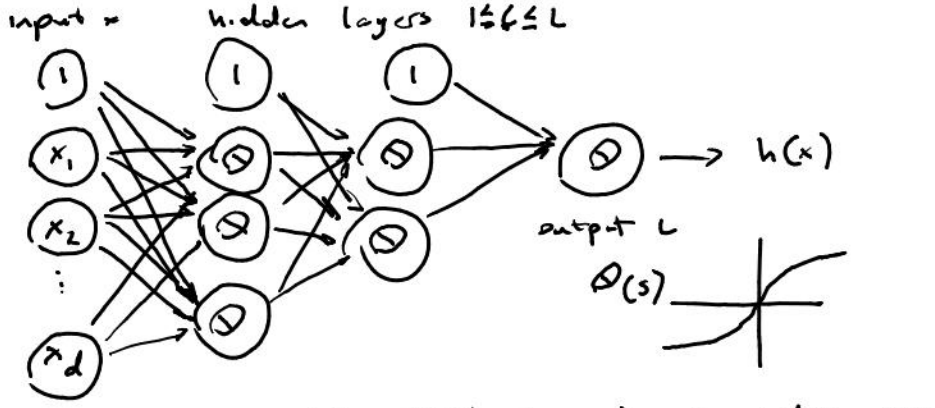


8 perceptrons

2 red flags — obv generalisation

also optimisation
↑ difficult to solve ☹

① soften threshold to make error surface smooth



in practice not all nodes use same θ

How π operates

$\theta(s) = \tanh(s) = \dfrac{e^s - e^{-s}}{e^s + e^{-s}}$ (looks linear near 0, hard near 1/-1)

$w_{ij}^{(\ell)}$ $\begin{cases} 1 \le \ell \le L & \text{layers} \\ 0 \le i \le d^{(\ell-1)} & \text{inputs} \\ 1 \le j \le d^{(\ell)} & \text{outputs} \end{cases}$

↖ DIMENSION OF LAYER

$$x_j^{(\ell)} = \theta(s_j^{(\ell)}) = \theta\left(\sum_{i=0}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} \right)$$

APPLY x to $x_1^{(0)} \dots x_{d^{(0)}}^{(0)} \rightarrow \rightarrow \rightarrow x_1^{(L)} = h(x)$

How do we find the weights?

BACKPROPEGATION! APPLY SGD

All weights $w = \{w_{ij}^{(\ell)}\}$ determine $h(x)$

Error on example (x_n, y_n) is $e(h(x_n), y_n) = e(w)$

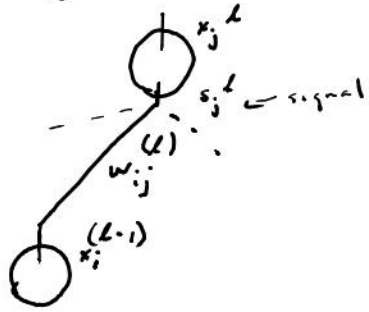
need $\nabla e(w)$: $\dfrac{\partial e(w)}{\partial w_{ij}^{(\ell)}}$ for all i, j, ℓ

we need to do this <u>efficiently</u> - FFT mentioned

COMPUTING $\partial e(w) / \partial w_{ij}^{(l)}$

we can evaluate one by one: analytically or numerically

but would prefer all at once!



trick: $\dfrac{\partial e(w)}{\partial w_{ij}^{(l)}} = \dfrac{\partial e(w)}{\partial s_j^{(l)}} \times \dfrac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}$

chain rule for partials work because weights are isolated to layer signal

we have $\dfrac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = x_i^{(l-1)}$

we only need $\dfrac{\partial e(w)}{\partial s_j^{(l)}} = \delta_j^{(l)}$ ← TRICKIER

δ for last layer ← start here and back-propagate

$\delta_j^{(l)} = \dfrac{\partial e(w)}{\partial s_j^{(l)}} \Rightarrow \delta_1^{(L)} = \dfrac{\partial e(w)}{\partial s_1^{(L)}}$; $e(w) = e(h(x_n), y_n)$

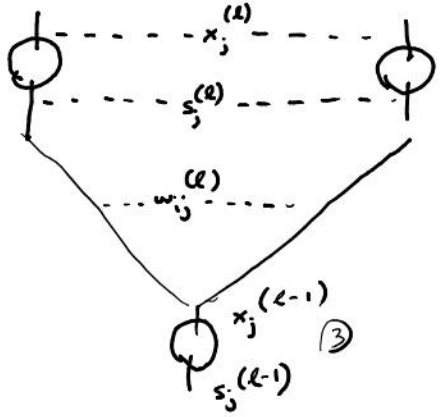
$h(x_n) = x_1^{(L)}$

eg $e(w) = (x_1^{(L)} - y_n)^2$ ← for squared error

$x_1^{(L)} = \theta(s_1^{(L)})$

for tanh → $\theta'(s) = 1 - \theta^2(s)$

BACK PROPEGATE δ



$$\delta_i^{(l-1)} = \partial e(w) / \partial s_i^{(l-1)}$$

$$= \sum_{j=1}^{d^{(l)}} \partial e(w) / \partial s_j^{(l)} \times \partial s_j^{(l)} / \partial x_i^{(l-1)} \times \partial x_i^{(l-1)} / \partial s_i^{(l-1)}$$

(1) (2) (3)

$$= \sum_{j=1}^{d^{(l)}} (1) \times (2) \times (3)$$

$$(3) = \theta'(s_i^{(l-1)})$$

$$(2) = w_{ij}^{(l)}$$

$$(1) = \delta_j^{(l)}$$

$$\delta^{(l-1)} = (1 - (x_i^{(l-1)})^2) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)} \delta_j^{(l)}$$

ALGORITHM

1: INITIALISE ALL WEIGHTS $w_{ij}^{(l)}$

2: for $t = 0, 1, 2, \ldots$ do

3: Pick $n \in \{1, 2, \ldots, N\}$

4: FORWARD: COMPUTE ALL $x_j^{(l)}$

5: BACKWORD: " $\delta_j^{(l)}$

6: UPDATE WEIGHTS: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta\, x_i^{(l-1)}\, \delta_j^{(l)}$

7: ITERATE UNTIL TERMINATION

8: RETURN

① DON'T INITIALISE TO ZERO; DO IT RANDOMLY

↓

MATH BREAKS AND CAN'T ADJUSTS, RANDOM BREAKS SYMMETRY

FINAL REMARK: HIDDEN LAYERS

nonlinear transform (learned, not imposed)

$d_{vc} \sim$ # weights

interpretation? Hard :> explanation is not what we're optimising

HOMEWORK #5

1. $\overbrace{E_D[E_{in}(w_{lin})]}^{0.08} = \sigma^2\left(1 - \frac{d+1}{N}\right)$

$\sigma = 0.1$

$d = 8$

$\hookrightarrow 0.088 = 0.01\left(1 - \frac{9}{N}\right)$

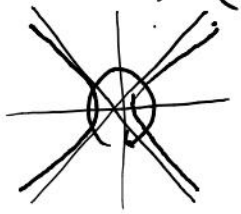
$0.8 = 1 - \frac{9}{N}$

$\frac{9}{N} = 0.2$

$N = 45$

$\therefore$ c ✓

2. $\Phi(1, x_1, x_2) = (1, x_1^2, x_2^2)$



weights have to be different signs, operating in Z-space

experiment: $x_1 = 0$, $x_2 = 1 \Rightarrow +1$

$0 + 1 = 1$

$x_1 = -15$, $x_2 = 0 \rightarrow -1$

$(-)225 \rightarrow -1$ ✓

$\therefore$ d ✓

3. d_{VC} of $\underline{\Phi}_4 : x \rightarrow (1, x_1, x_2 \ldots x_2^4) = 15$

$\therefore$ c ✓

4. $E(u,v) = \overbrace{(ue^v - 2ve^{-u})^2}^{\alpha}$

$= 2\alpha$

$\alpha = \left(e^v + 2ve^{-u}\right)\left(ue^v - 2ve^{-u}\right)$

e ✓

5. WRITE A PROGRAM IMPLEMENTING GRADIENT DESCENT ON THIS ERROR SURFACE :)

LEARNING RATE = 0.1; starting point is (1, 1)

$\partial u/\partial x = 2\left(e^v + 2ve^{-u}\right)\left(ue^v - 2ve^{-u}\right)$

$\partial v/\partial x = 2\left(ue^v - 2e^{-u}\right)\left(ue^v - 2ve^{-u}\right)$

COUNT FOR $(e^{-u}$ THRESHOLD = 10 d ✓

6. $[u,v] = [0.045, 0.023] = c$ ✓

7. error on coordinate descent = 0.13 → a ✓

8. WRITE A PROGRAM IMPLEMENTING LOGISTIC REGRESSION - NB THIS IS A WEIRD PERCEPTRON!

REPEAT 100x w/ $w^{(0)} = [0, 0, 0]$

TERMINATE WHEN $\|w^{(t+1)} - w^{(t)}\| < 0.01$

CROSS-ENTROPY ERROR =

8. $E_{out} = 0.104 \rightarrow$ d ✓

9. Epochs : 330 → a ✓

10. PLA as SGD

PLA error is $\text{sign}(y w^T x_n)$

not a nice gradient → e ✓

LECTURE 11: OVERFITTING

(1) • what is overfitting? applies to every machine learning problem; tools to deal with it are universal

(2) • the role of noise

(3) • deterministic noise

(4) • dealing with overfitting

(1) AN ILLUSTRATION OF OVERFITTING

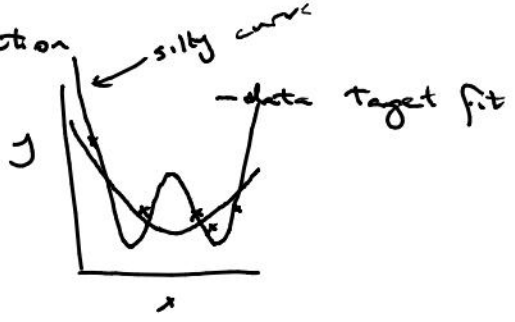
simple target function

five data points

small noise

let's fit w/ 4th order

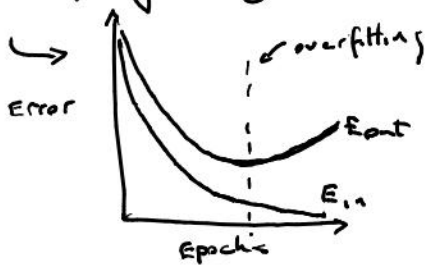
oh no!!



$E_{in} = 0$, E_{out} is huge

OVERFITTING WITHIN SAME MODEL

· neural net fitting noisy data



as you explore more of the weight space generalisation error gets worse

overfitting occurs when E_{in} ↓ corresponds to E_{out} ↑

OVERFITTING: FITTING THE DATA MORE THAN IS WARRANTED

CULPRIT: FITTING THE NOISE - HARMFUL; EXTRAPOLATES NOISE OUT OF SAMPLE "HALLUCINATION"

CASE STUDY TENTH ORDER TARGET + NOISE

↓VS

50TH ORDER TARGET w/o NOISE

TWO FITS FOR EACH TARGET

(α)

	2^{nd}	10^{th}
E_{in}	0.05	0.034
E_{out}	0.127	9.00

(β)

	2^{nd}	10^{th}
E_{in}	0.27	~0
E_{out}	0.12	7689 ← lol

↑ w/c got overfitting w/o noise!

TWO LEARNERS LEARNING A 10^{TH} - ORDER TARGET

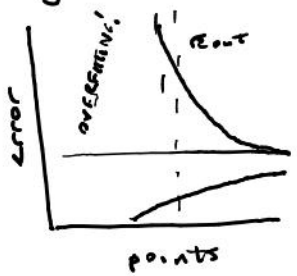
O - 10^{TH} ORDER MODEL

R - 2^{ND} ORDER MODEL "HOW MANY POINTS DO WE HAVE?"

THE DATA MATTERS MORE THAN THE TARGET FUNCTION. MATCH THE RESOURCES!

learning curves

O



R

- loses w/ inadequate data resources

what about our noise-free overfitting?

$\mathcal{H}_{10}$ & $\mathcal{H}_2$ learning 50th order target. no noise!

but is there really no noise?

A DETAILED EXPERIMENT

Impact of noise level & target complexity

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

$f(x)$: target

$\epsilon(x)$: noise (i.i.d. ind. rv) $\sim \sigma^2$

$$y = \underbrace{\sum_{q=0}^{Q_f} a_q x^q}_{\text{normalized}} + \epsilon(x)$$

← letting us pick $f(x)$ of arbitrary power

& not random - use Legendre polynomials so we can access more of the target space

	for 10^{th} order
noise level: σ^2	
target complexity: Q_f	$Q_f = 10$
data set size: N	$N = 15$
	$\sigma^2 = ?$

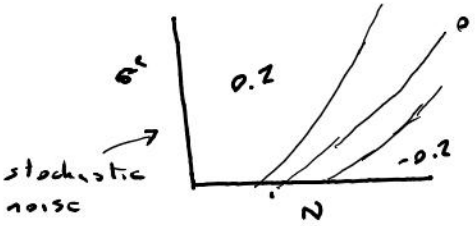
THE OVERFIT MEASURE

we fit the dataset $(x_1, y_1), \ldots, (x_N, y_N)$ using our two models: $\mathcal{H}_2$ & $\mathcal{H}_{10}$.

compare E_{out} of $g_2 \in \mathcal{H}_2$ & $g_{10} \in \mathcal{H}_{10}$

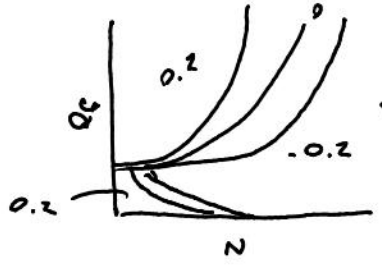
overfit measure = $E_{out}(g_{10}) - E_{out}(g_2)$

LETS look at $10e^7$ ITERATIONS



positive → high overfitting

as we increase noise, overfitting gets worse



Q_f seems entangled with N

what can we derive from this?

overfiting not determined by conventional noise alone

number of points ↑ overfitting ↓

stochastic noise ↑ " ↑

deterministic " ↑ " ↑

DEFINITION OF DETERMINISTIC NOISE

THE PART OF f THAT $\mathcal{H}$ CANNOT CAPTURE

THE NOISE IN THE LEARNING VS THE NOISE IN THE DATA?

DETERMINISTIC NOISE 1. DEPENDS ON $\mathcal{H}$

2. fixed for given x

neither is true for stochastic noise

IMPACT ON OVERFITTING

deterministic noise $\uparrow$ Q_f ← kicks in when $Q_c < d_{vc}$

for a finite sample you <u>will</u> model noise

noise $\uparrow$ bias-variance

decomposition $\mathbb{E}_D\left[(g^{(D)}(x) - f(x))^2\right] = \text{var}(x) + \text{bias}(x)$

if f is noisy?

$y = f(x) + \epsilon(x)$; $\mathbb{E}[\epsilon(x)] = 0$

$$\mathbb{E}_{D,\epsilon}\left[(g^{(D)}(x) - y)^2\right] = \mathbb{E}_{D,\epsilon}\left[(g^{(D)}(x) - f(x) - \epsilon(x))^2\right]$$

$$\Downarrow$$

$$\text{var}(x) + \text{bias}(x) + \mathbb{E}_{\epsilon,x}\left[(\epsilon(x))^2\right]$$

DETERMINISTIC

THERE IS NOISE IN THE BIAS :)

AND THE BIAS IS INDEPENDENT OF N!

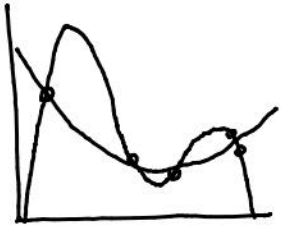
Bias ξ σ^2 push variance around

(4) DEALING WITH OVERFITTING (3)

TWO CURES: REGULARISATION - hit the brakes

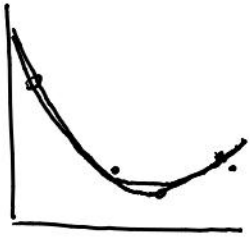
VALIDATION - check the bottom line

(1)



free fit

|| regularisation restraint/friction; 1%



← much better

LECTURE 12: REGULARISATION

REMEMBER: d_{vc} ANALYSIS ALLOWS FOR OVERFITTING; IT DOESN'T PREDICT IT. THE VC BOUND IS FOR ALL POSSIBLE TARGETS

DETERMINISTIC NOISE IS A FUNCTION OF THE LIMITATIONS OF YOUR MODEL - CANNOT LEARN OUTSIDE THE CAPACITY OF YOUR HYPOTHESIS SET. BIAS

REGULARISATION - USE IN EVERY ML APPLICATION

- INFORMAL (1)
- FORMAL (2)
- WEIGHT DECAY (3)
- CHOOSING REGULARISER (4)

TWO APPROACHES -

MATHMATICAL

ILL-POSED PROBLEMS IN FUNCTION APPROXIMATION

IF BAYES USED, THIS GOES IN THE PRIOR

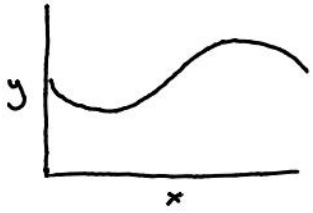
BUT IN MOST CASES THE ASSUMPTIONS HERE DON'T HOLD!

BEST UTILITY IS TO DEVELOP MATHEMATICAL INTUITION FOR REAL-WORLD CASES

HEURISTIC

HANDCAPPING WEIGHTS, FRICTION, PURELY INTUITIVE

EXAMPLE



instead of crazy lines like before, we restrict weights. lose perfect E_{in} but gain E_{out}

w/o REG	w/ REG
bias: 0.21	bias: 0.23
variance: 1.69	variance: 0.33

REGULARISATION REDUCES VARIANCE BUT MAY BUMP BIAS - WE'RE KILLING SOME SIGNAL AS WELL AS NOISE. BUT THE GAIN IN VARIANCE IS WORTH THE TRADE

THE POLYNOMIAL MODEL

WE USE LEGENDRE (α) POLYNOMIALS $L_1 \dots L_Q$

MUTUALLY ORTHOGONAL*

$\mathcal{H}_Q$: polynomials of order Q

$$z = \begin{bmatrix} 1 \\ L_1(x) \\ \vdots \\ L_Q(x) \end{bmatrix} \qquad \mathcal{H}_Q = \left\{ \sum_{q=0}^{Q} w_q L_q(x) \right.$$

α $L_1 = x$

$L_2 = \frac{1}{2}(3x^2 - 1)$

$L_3 = \frac{1}{2}(5x^3 - 3x)$

$L_4 = \frac{1}{8}(35x^4 - 30x^2 + 3)$

$L_5 = \frac{1}{8}(63x^5 \dots)$

apply linear regression in Z-space

REMEMBER - TARGET FUNCTION UNKNOWN!

UNCONSTRAINED SOLUTION

given $(x_1, y_1), \ldots, (x_N, y_N) \rightarrow (z_1, y_1), \ldots, (z_N, y_N)$

MINIMISE $E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} (w^T z_n - y_n)^2$

$$= \frac{1}{N} (Zw - Y)^T (Zw - Y)$$

$$Z = \begin{bmatrix} - z_1 - \\ \vdots \\ - z_N - \end{bmatrix}; \quad Y = \begin{bmatrix} - y_1 - \\ \vdots \\ - y_N - \end{bmatrix}$$

$$w_{lin} = (Z^T Z)^{-1} Z^T Y$$

CONSTRAIN THE WEIGHTS

hard constraint: $\mathcal{H}_2$ is constrained version of $\mathcal{H}_{10}$ ($w_q = 0$ for $q > 2$)

softer version: $\sum_{q=0}^{Q} w_q^2 \leq C$ "soft-order" constraint; $\mathcal{H}_2$ is "budget" subset of $\mathcal{H}_Q$

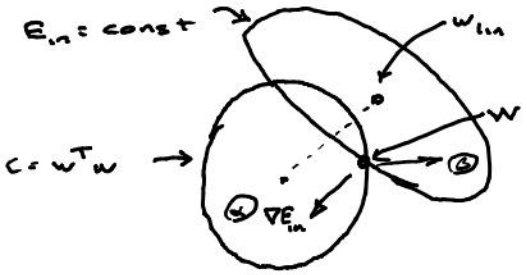
this constraint pushes weights to be small; it doesn't force any to zero

MINIMISE: $\frac{1}{N} (Zw - Y)^T (Zw - Y) = w_{reg}$ (not w_{lin})

subject to $w^T w \leq C$

SOLVING FOR w_{reg}

MINIMISE $E_{in}(w) = \frac{1}{N}(Zw - Y)^T(Zw - Y)$; $w^T w \leq C$



if C large enough to encompass w_{lin}; answer is w_{lin}

(1) ∇ orthogonal to E_{in} surface

(2) normal to $w^T w = C$; w_{reg}

I reckon this is $\nabla E_{in} - w = 0$ ✓

$$\nabla E_{in}(w_{reg}) \propto -w_{reg}$$

$$= -2\frac{\lambda}{N} w_{reg}$$ ← reformatted for convenience (checks out)

$$\nabla E_{in}(w_{reg}) + 2\frac{\lambda}{N} w_{reg}$$

MINIMISE $E_{in}(w) + \frac{\lambda}{N} w^T w$

↑ regularisation term

λ is dependent of C, dataset, other stuff - but we know one exists

$C \uparrow \quad \lambda \downarrow$

AUGMENTED ERROR

$$E_{aug}(w) = \frac{1}{N}(Zw-Y)^T(Zw-Y) + \frac{\lambda}{N}w^Tw \quad \text{unconditionally}$$

— solves —

$$E_{in}(w) = \frac{1}{N}(Zw-Y)^T(Zw-Y) \quad \text{subject to } C \geq w^Tw$$

SOLUTION!

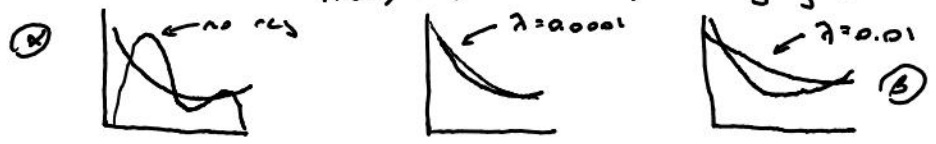
$$\nabla E_{aug}(w) = 0 \Rightarrow Z^T(Zw-Y) + \lambda w = 0$$

$$w_{reg} = (Z^TZ + \lambda I)^{-1}Z^TY$$

vs. $w_{lin} = (Z^TZ)^{-1}Z^TY$

RESULT

MINIMISING $E_{in}(w) + \lambda/N\, w^Tw$ for varying λ



too much λ cares more about driving weights down than fitting data, choice of λ is critical!

(α) OVERFITTING

(β) UNDERFITTING

WEIGHT 'DECAY'

minimising $E_{in}(w) + \frac{\lambda}{N} w^T w$ is called weight decay. why?

GRADIENT DESCENT: $w(t+1) = w(t) - \eta \nabla E_{in}(w(t)) - 2\eta \frac{\lambda}{N} w(t)$

$= w(t)\left(1 - 2\eta \frac{\lambda}{N}\right) - \eta \nabla E_{in} w(t)$

↑ deflects step towards origin

VARIATIONS OF WEIGHT DECAY

EMPHASIS ON CERTAIN WEIGHTS: $\sum_{q=0}^{Q} \gamma_q w_q^2$

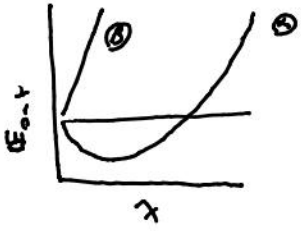
$\gamma_q = 2^q \Rightarrow$ low order fit

$\gamma_q = 2^{-q} \Rightarrow$ high order fit

neural nets get different γ in different layer

TIKHONOV REGULARISER: $w^T \Gamma^T \Gamma w$ (proper choice of Γ gives you weight decay etc.)

EVEN WEIGHT GROWTH!



(B) WEIGHT DECAY

(A) WEIGHT GROWTH

MAYBE DON'T CHOOSE WEIGHT GROWTH

PRACTICAL RULE: stochastic noise is high-frequency

deterministic noise is non-smooth

THEREFORE => constrain learning towards smoother hypothesis

GENERAL FORM OF AUGMENTED ERROR

regulariser $\Omega = \Omega(h)$, minimise $E_{AUG}(h) = E_{in}(h) + \frac{\lambda}{N}\Omega(h)$

- similar to $E_{out}(h) \leq E_{in}(h) + \Omega(h)$ (*)

we're using E_{aug} to estimate E_{out} better than E_{in} can

(*) GENERALISATION

CHOOSING A REGULARISER

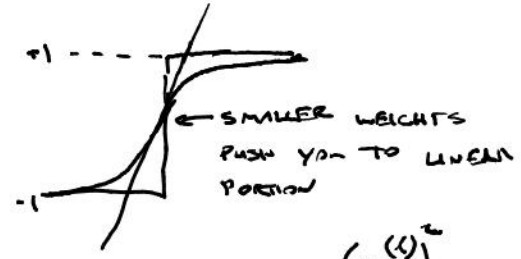
'PERFECT' Ω :
- CONSTRAINS IN DIRECTION OF $f(x)$
- HARM NOISE MORE THAN SIGNAL
- GENERALLY WANTS TO BE SMOOTHER

WHAT IF WE PICK BAD Ω? IT'S OK, WE HAVE λ :)

WE HAVE TO USE VALIDATION TO CHECK

NN Ωs : WEIGHT DECAY

from linear to logical

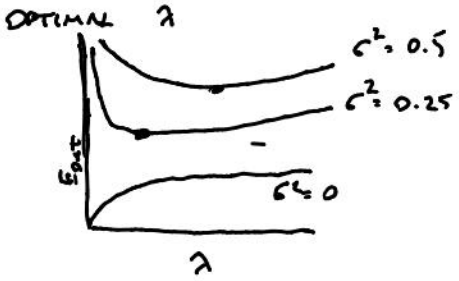


WEIGHT ELIMINATION

SMALLER d_{VC}! soft weight elimination $\Omega(w) = \sum_{i,j,l} \frac{\left(w_{ij}^{(l)}\right)^2}{\beta^2 + \left(w_{ij}^{(l)}\right)^2}$

EARLY STOPPING - regularisation through the optimiser!

(when to stop?!) BEST TO SEPARATE CONCERNS :)



Stochastic ← but also applies to Q_f!!

DETERMINISTIC NOISE IN OVERFITTING BEHAVES ALMOST EXACTLY LIKE STOCHASTIC NOISE

HOMEWORK #6

1. DERMINISTIC NOISE $\propto$ $\mathcal{H}$

   $\mathcal{H}' \subset \mathcal{H}$, f is fixed. this means $\mathcal{H}'$ is less powerful than $\mathcal{H}$, so b ✓

2. REGULARISATION w/ WEIGHT DECAY

   (REMEMBER FORM: $(Z^T Z - \lambda I)^{-1} Z Y = w$

   $\phi(x_1, x_2) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, |x_1 - x_2|, |x_1 + x_2|)$

   w/o DECAY: $E_{in} \sim 0.3$, $E_{out} \sim 0.8$ => a ✓

3. w/ $k=-3$: " => d ✓

4. w/ $k=3$: $E_{in} = 0.37$, $E_{out} = 0.43$ => c ✓

5. best k: -1 => d ✓

6. $E_{out} = 0.036$ => b ✓

7. $\mathcal{H}(Q_1, C, Q_0)$ ← stopping
constraint ↑ ← max degree = 1

$$\mathcal{H}_Q : \{h \mid h(x) = w^T z = \sum_{q=0}^{Q} w_q L_q(x)\}$$

a) $\underbrace{\mathcal{H}(10,0,3)}_{\mathcal{H}_2} \cup \underbrace{\mathcal{H}(10,0,4)}_{\mathcal{H}_2} \Rightarrow \mathcal{H}_5$

b) $\underbrace{\mathcal{H}(10,0,3) \cup \mathcal{H}(10,1,4)}_{\text{BOTH WEIRD, } \neq \mathcal{H}_3}$

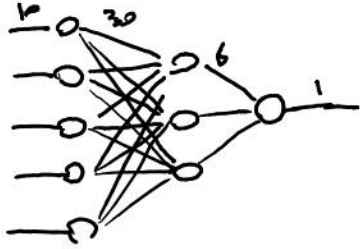
c) $\underbrace{\mathcal{H}(10,0,3)}_{\mathcal{H}_2} \cap \underbrace{\mathcal{H}(10,0,4)}_{\mathcal{H}_3} \Rightarrow \mathcal{H}_2$ ✓

⋮

8. NEURAL NETWORK - fully connected

$L = 2$; $d^{(0)} = 5$, $d^{(1)} = 3$, $d^{(2)} = 1$

what does this look like?



47? c ✓

NOTE: EACH HIDDEN LAYER ALSO RECIEVES A BIAS WEIGHT. SO

$w = 10a_1 + (a_1+1)a_2 + \ldots + (a_{n-1}+1)a_n$

min 2 nodes per hidden layer.

$\therefore$ min weights is $n = 17$; 1 computing node / layer

$w = 10 \cdot 2 + \ldots + 2 \cdot 1$

$20 + 17 \cdot 2 + 2 \Rightarrow 46$ a ✓

Maximum would be $n = 2$, 2 hidden nodes & 34 compute

$10a_1 + (a_1+1)a_2 + (a_2+1)$

$10a_1 + a_1a_2 + 2a_1 + 1$

$10a_1 + a_1(34-a_1) + 2(34-a_1) + 1$

$10a_1 + 34a_1 - a_1^2 + 68 - 2a_1 + 1$

$42a_1 - a_1^2 + 69 = 0$

$-2a_1 + 42 = 0 \Rightarrow a_1 = 21, a_2 = 13$

$210 + 286 + 14 = 510 \Rightarrow$ e ✓

LECTURE 13: VALIDATION

- TO SOLVE λ in regularisation we need to take the lagrangian
- remember E_{AUG} is a better proxy than E_{in}
- TODAY WE'LL FOCUS ON CHOOSING λ in

$$E_{AUG}(h) = E_{in}(h) + \lambda/N\,\Omega(h) - \text{validation}$$

VALIDATION

- used in pretty much every machine learning application
- the validation set
- model selection
- cross-validation

VALIDATION VS. REGULARISATION

$E_{out}(h) = E_{in}(h) + \text{penalty}$

regularisation: estimates overfit penalty

validation: estimates $E_{out}(h)$

Analysing the estimate

on <u>o-o-s</u> point (x, y), the error is $e(h(x), y)$

squared: $(h(x)-y)^2$

binary: $[\![h(x) \neq y]\!]$

$\mathbb{E}\left[e(h(x), y)\right] = E_{out}(h)$

← unbiased error (but not necessarily accurate)

$\text{var}\left[e(h(x), y)\right] = \sigma^2$ ← estimate on one point

FROM POINT TO SET

on a validation set $(x_1, y_1), \ldots, (x_K, y_K)$, ← K = size of validation set

the error $E_{val}(h) = \frac{1}{K} \sum_{k=1}^{K} e(h(x_k), y_k)$

$\mathbb{E}\left[E_{val}(h)\right] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}\left[e(h(x_k), y_k)\right] = E_{out}(h)$

we need to improve variance — does going from point to set help us?

$\text{var}\left[E_{val}(h)\right] = \frac{1}{K^2} \sum_{k=1}^{K} \text{var}\left[e(h(x_k), y_k)\right] = \frac{\sigma^2}{K}$ ← BETTER!

↖ covariants drop out, just use diagonals

estimate reliability depends on K $\quad E_{val}(h) = E_{out}(h) \pm O\left(\frac{1}{\sqrt{K}}\right)$

K is not free — we separate it out of the training set, so given data set

$D = (x_1, y_1), \ldots, (x_N, y_N)$

D_{val} { K points → validation, chosen randomly

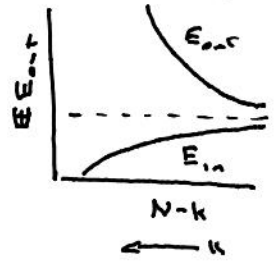
N-K points → training

D_{train}

$D_{val} \cup D_{Train} = D$

$O\left(\frac{1}{\sqrt{K}}\right)$: small K => bad estimate

large K => generalisation cost



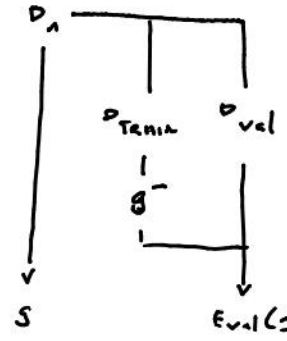
getting a better estimate of worse quantity!

IS THERE A TRICK TO AVOID THIS?

TAKE K, THEN PUT BACK INTO D_{train}

$D \to D_{TRAIN} + D_{val}$

$D \Rightarrow g \quad D_{train} \Rightarrow g^-$



is $E_{val}(g^-)$ ~ $E_{val}(g)$?

$E_{val} = E_{val}(g_j^-)$ Large K => bad estimate!

so how do we pick K?

rule of thumb: $K = N/5$

WHY IS THIS CALLED VALIDATION?

- we use it to make choices
- if an estimate of E_{out} affects learning the set is no longer a test set! it's a validation set

WHAT'S THE DIFFERENCE?

TEST SET IS UNBIASED; VALIDATION SET HAS OPTIMISTIC BIAS. TWO HYPOTHESES, h_1 & h_2 w/ $E_{out} = 0.5$

estimate error e_1, e_2 uniform on $[0,1]$ (expected error 0.5)

pick $h \in \{h_1, h_2\}$ with $e = \min(e_1, e_2)$

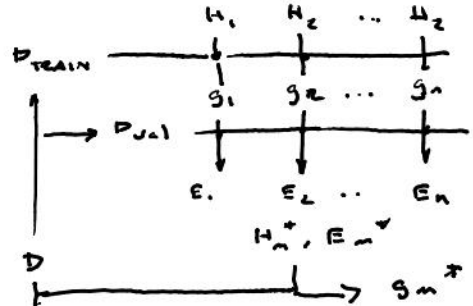
$E(e) < 0.5$ ← optimistic bias

USING D_{VAL} MORE THAN ONCE

M models $\mathcal{H}_1, \ldots, \mathcal{H}_M$

use D_{TRAIN} to learn g_m^-

EVALUATE g_m^- USING D_{VAL}



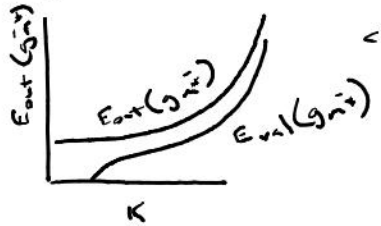
THIS ISN'T DATA SNOOPING BECAUSE IT'S PART OF THE LEARNING PROCESS? AND HOW DO WE CONTROL BIAS?

$E_m = E_{val}(g_m^-); \quad m = 1, \ldots, M$

pick model $m = m^*$

THE BIAS

WE SELECTED $\mathcal{H}_{m^*}$ USING D_{VAL}



← selecting between 2 $\mathcal{H}$ can a lot

WHY IS E_{out} GOING UP?! BIGGER K → SMALLER N

WHY ARE THE CURVES CONVERGING? BIGGER K → BETTER ESTIMATE

FOR M models: $\mathcal{H}_1, \ldots, \mathcal{H}_M$ D_{val} IS USED FOR TRAINING ON THE FINALISTS MODEL:

$\mathcal{H}_{VAL} = \{g_{VAL1}^-, g_{VAL2}^-, \ldots, g_{VALM}^-\}$

BACK TO HOEFDING AND VC

$$E_{out}(g_{m^*}^-) \leq E_{VAL}(g_{m^*}^-) + O\left(\sqrt{\frac{\ln(M)}{K}}\right)$$

CAN M BE INFINITE? YES!

IS SIMPLE PARAMETER w/ VC dimension = 1!

regularisation parameter λ; early-stopping T

THE MORE PARAMS WE ADD TO VALIDATION THE LESS ACCURATE

DATA CONTAMINATION

ERROR ESTIMATES: $E_{in}, E_{test}, E_{val}$

CONTAMINATION: OPTIMISTIC (DECEPTIVE) BIAS IN ESTIMATING E_{out}

TRAINING SET: totally contaminated

TEST " : totally 'clean'

VALIDATION " : slightly contaminated :)
(multiple sets!?)

CROSS - VALIDATION

$$E_{out}(g) \approx E_{out}(g^-) \approx E_{val}(g^-)$$
(small k) (large k)

HOW DO WE DO BOTH?

"LEAVE ONE OUT"

USE N-1 POINTS FOR TRAINING; 1 POINT FOR VALIDATION

$D_n = (x_1, y_n), \ldots, (x_{n-1}, y_{n-1}), \cancel{(x_n, y_n)}, (x_{n+1}, y_{n+1}), \ldots, (x_N, y_N)$

final hypothesis learned from D_n is g_n^-

$e_n = E_{val}(g_n^-) = e(g_n^-(x_n), y_n)$

WHAT IF WE DO THIS ACROSS ALL POINTS?

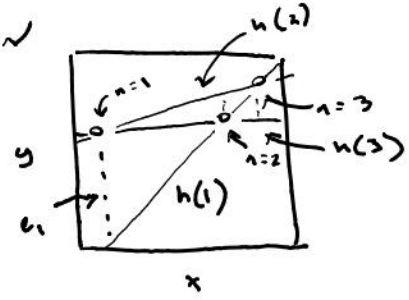
HYPOTHESES ARE ALL DIFFERENT BUT ALL TRAINED ON N-1 POINTS

WE KNOW THEY'RE RELATED wrt E_{out}

cross-validation error: $E_{cv} = \frac{1}{N}\sum_{n=1}^{N} e_n$ ← NOT INDEPENDENT BUT SEEMS NOT TO MATTER :)

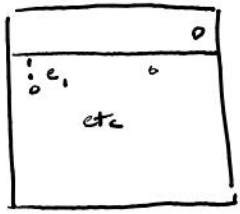
ILLUSTRATION

Linear:



$E_{cv} = \frac{1}{3}(e_1 + e_2 + e_3)$

constant



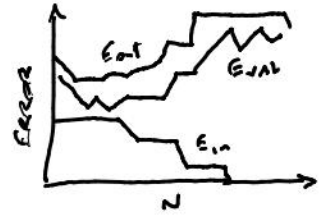
check E_{cv} and see which model is better

CROSS-VALIDATION IN ACTION

digit-classification requires non-linear transform

$(1, x_1, x_2) \rightarrow$ something giant

use cross-validation to choose which features to use!



w/o VALIDATION:

$E_{in} = 0\%$, $E_{out} = 2.5\%$

w/ VALIDATION

$E_{in} = 0.8\%$, $E_{out} = 1.5\%$

GENERALLY... LEAVE MORE THAN ONE OUT

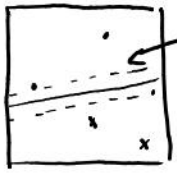
So we only run 8 times :) N/k on $N-k$ points each. In practice 10-fold cross validation works well

LECTURE 14 : SUPPORT VECTOR MACHINES

- MOST SUCCESSFUL CLASSIFICATION TOOL IN ML
  - MAXIMISING THE MARGIN
  - THE SOLUTION
  - NONLINEAR TRANSFORMS

BETER LINEAR SEPARATION

DATA



many possible lines to separate the data. is there an advantage in choosing any particular one?

imagine ---- ↕ as the margin. this is a measure of robustness to noise! so we ① maximise margin.

which ② maximises the margin?

① REMEMBER THE GROWTH FUNCTION?

- CHOOSING BIG MARGIN CONSTRAINS DICHOTOMIES!

∴ SEPARATING POINTS WITH LARGE MARGINS DROPS D_{VC}!

② FINDING w

LET x_n BE THE NEAREST DATA POINT TO THE PLANE $w^T x = 0$. DISTANCE IS MARGIN

2 PRELIMINARY TECHNICALITIES:

1. NORMALISE w:

$|w^T x_n| > 0 \longrightarrow |w^T x_n| = 1$

i. w^T has a scale invariance but non-unit w will cause problems later

ii. expanding $w^T x_n$ currently gives a 'distance' but it's not Euclidean; normalising gives us a proper distance

2. Pull out w_0 ← no longer convenient to use, so $w = (w_1, \ldots, w_d)$ apart from 1

The plane is now $\boxed{w^T x + b = 0}$ (no x_0)

COMPUTING THE DISTANCE

THE DISTANCE BETWEEN X_n and plane $w^T x + b = 0$ WHERE $|w^T x_n + b| = 1$.

$w \perp$ TO THE PLANE IN X SPACE

TAKE x' & x'' on plane

$w^T x' + b = 0$, $w^T x'' + b = 0 \Rightarrow w^T(x' - x'') = 0 \Rightarrow w^T$ orthogonal

and the distance is...

take any point on the plane. take the projection of $x_n - x$ on w

$$\hat{w} = \frac{w}{\|w\|} \Rightarrow \text{distance} = \left| \hat{w}^T (x_n - x) \right|$$

$$\text{distance} = \frac{1}{\|w\|} \left| w^T x_n + b - w^T x - b \right|$$

$$\Downarrow$$

$$\frac{1}{\|w\|}$$

what are the bs doing? making $w^T x$ terms go away!

② subject to $\min_{n=1,2,\dots,N} |w^T x_n + b| = 1$

notice: optimising $\min$ = :c

$$|w^T x_n + b| \overset{①}{=} y_n (w^T x_n + b)$$

① SIGNAL AGREES WITH VALUE

EQUIVALENT TO

$$\text{minimise } \frac{1}{2} w^T w$$

② subject to $y_n (w^T x_n + b) \geq 1$

for $n = 1, 2, \dots, N$

③ THESE ARE EQUIVALENT BUT IT'S MESSY!

CONSTRAINED OPTIMISATION

MINIMISE $\frac{1}{2} w^T w$

SUBJECT TO $y_n(w^T x_n + b) \geq 1$ for $n = 1, 2, \ldots, N$

$w \in \mathbb{R}^d$, $b \in \mathbb{R}$

LAGRANGE? inequality constraints => KKT

WE'VE SEEN THIS BEFORE WITH : REGULARISATION

MINIMISE $E_{in}(w) = \frac{1}{N} (Zw - y)^T (Zw - y)$

subject to $w^T w \leq C$

∇E_{in} normal to constraint

	OPTIMISE	CONSTRAIN
REGULARISATION :	E_{in}	$w^T w$
SUM :	$w^T w$	E_{in} <- LOL

LAGRANGE FORMULATION

$$\min \rightarrow \mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n \left(y_n (w^T x_n + b) - 1 \right)$$ (slack)

wrt w & b and maximise w.r.t. each $\alpha_n \geq 0$

$$\nabla_w \mathcal{L} = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \quad \leftarrow \text{conditions}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

Substitute conditions into original Lagrangian

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1)$$

(① b multiplied by $-\sum_{n=1}^{N} \alpha_n y_n = 0$, so goes away)

$$\tilde{\mathcal{L}}(a) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m a_n a_m x_n^T x_m$$

maximise wrt α subject to $\alpha_n \geq 0$ for $n = 1, \ldots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

SOLUTION: QUADRATIC PROGRAMMING

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

$$\Downarrow$$

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$$

$$\min_{\alpha} \frac{1}{2}\alpha^T \underbrace{\begin{bmatrix} y_1 y_1 x_1^T x_1 & \cdots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & & \vdots \\ \cdots & & \\ y_N y_1 x_N^T x & \cdots & y_N y_N x_N^T x_N \end{bmatrix}}_{\text{quadratic coefficients } Q} \alpha + (-1^T)\alpha$$

subject to $\underbrace{y^T\alpha = 0}_{\text{linear constraint}}$

$$0 \le \alpha \le \infty$$

WARNING: Q is N×N ← uh oh! many heuristics to deal with big N

QP hands us α

$$\alpha = \alpha_1, \ldots, \alpha_N$$

$$\Rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

KKT condition: FOR $n = 1, \ldots, N$

$$\alpha_n(y_n(w^T x_n + b) - 1) = 0$$

EITHER SLACK IS ZERO OR α IS ZERO

FOR EACH POINT. INTERIOR POINTS HAVE $\alpha = 0$

$\alpha_n > 0 \Rightarrow x_n$ is a <u>support vector</u>

SUPPORT VECTORS

closest x_n's to plane: achieve the margin

$$\Rightarrow y_n(w^T x_n + b) = 1$$

$$w = \sum_{x_n \text{ is SV}} \alpha_n y_n x_n$$

α is model parameters, only taking support factors drops params a lot!

solve for b using any SV:

$$y_n(w^T x_n + b) = 1$$

NONLINEAR TRANSFORMS

SVMs give linear boundaries

transform $x \to z$?

what happens to SVM formulation?

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \underbrace{z_n^T z_m}_{\text{ONLY CHANGE}}$$

SVM optimisation is dimension agnostic!
(apart from inner product of $z_n^T z_m$)

"SUPPORT VECTORS" in X space

support vectors live in space you're analysing (Z space)

in X space, "pre-images" of support vectors

THE MARGIN EXISTS ONLY IN Z SPACE :>

GENERALISATION RESULT

$$\mathbb{E}[E_{out}] \leq \frac{\mathbb{E}[\#SV]}{N-1}$$

EXTRAS: DISTANCE È LAGRANGIAN

$$1/\|w\| \; |w^T(x_n - x)| \quad (1)$$

$w^T x + b = 0$ (x is on hyperplane)

$\therefore w^T x = -b$ (2)

EXPAND (1) $\Rightarrow w^T(x_n - x) \Rightarrow w^T x_n - w^T x$

SUBSTITUTE (2) $\Rightarrow w^T x_n + b$

BUT WE KNOW THAT $w^T x + b$ ARE FREELY SCALABLE, SO WE CAN SIMPLY CHOOSE TO SCALE IT SUCH THAT $w^T x_n + b = 1$; $\therefore$ DISTANCE $= \frac{1}{\|w\|}$

$$|w^T x_n + b| = y_n(w^T x_n + b) \leftarrow \text{for all correctly classified points}$$

replacing minimum over $n = 1$ w/ all N quantities are at least 1 is saying that each support vector is 1 and rescaling until we get there

LAGRANCE

- we have a constrained function
- build a new <u>unconstrained</u> function that encodes penalty term w/ a multiplier. so

$$L(x, a) = f(x) - a \cdot g(x) \quad \text{w/ } a \geq 0$$

then minimising $f(x)$ subject to $g(x) \geq 0$ is equivalent to finding a saddle point of L

we need to maximise $L(x,a)$ over $a \rightarrow M(x)$

then minimise $M(x)$ over x

$$\Rightarrow \min_x (\max_a (L(x, a))$$

FOR SVM LAGRANCIAN

$$L(w, b, a) = f(w, b) - \sum a_n (\text{constraint}_n) \quad a_n \geq 0$$

FIX SOME PARTICULAR (w, b). WHAT IS

$$\max_a (L(w, b, a))?$$

CASE 1: (w, b) satisfies constraint

↳ $-a_1 \text{constraint}_n$. set a_n to 0 whenever constraint is satisfied to maximise negative sum

CASE 2: (w, b) violates constraint

- a_n constraint$_n$ => positive, maximise by $a_n \Rightarrow \infty$

so $\max_a \left(L(w,b,a)\right) = f(w,b)$ when constraints are satisfied and ∞ when not.

this builds an infinite wall when we come to minimise x!

then back substitute as per your lecture notes

NOTE: THE VALUE OF THE LAGRANGE PENALTY, α/a TELLS US SOMETHING ABOUT WHERE THE SOLUTION WANTED TO GO. IF ZERO, NOT RELEVANT. FOR SVMs, ONLY NON-ZERO AT SUPPORT VECTORS

HOMEWORK #7

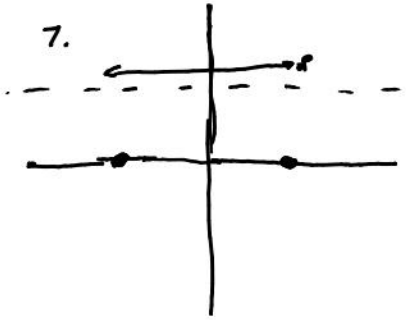
1. $k=6$ d ✓
2. $k=7$ e ✓
3. $k=6$ d ✓
4. $k=6$ d ✓
5. 0.1, 0.2 b ✓

6. e_1, e_2 independent random vars distributed over $[0,1]$ evenly. $\therefore \mathbb{E}\, e_1, e_2 = 0.5$

But what is $\mathbb{E} \min(e_1, e_2)$? $\Rightarrow 1/3$

$\therefore$ 0.5, 0.3, 0.4 $\Rightarrow d$

7.



what p is makes $h_0(x) = h_1(x)$? (constant; $ax+b$)

$h_0(x)$: 1. $-D_1 \rightarrow y = 1, 0, \frac{1}{2} \Rightarrow$ squared error $= 1/4$

2. $-D_2 \rightarrow y = 0, 0, 1 \Rightarrow$ " 1

3. $-D_3$ a> D_1

$\therefore E_{cv}(h_0) = \frac{1}{3}(1.5) = 1/2$

$h_1(x)$ 1. $-D_1 \rightarrow y = (x - 1/p - 1)$ & $D_3 = 4/(1-p)^2$

$x = -1; \quad y = -2/(1-p)$

$e_1 = \left(0 - 2/(1-p)\right)^2 = 4/(1-p)^2$

2. $-D_2 \rightarrow y = 0$

$x = p \Rightarrow y = 0$

$e_2 = 1$

$E_{cv}(h_1) = \frac{1}{3}\left(1 + 4/(p+1)^2 + 4/(p-1)^2\right)$

SET $E_{cv}(h_1) = E_{cv}(h_0)$

$$1/2 = 1/3\left(1 + 4/(\rho+1)^2 + 4/(1-\rho)\right)$$

$$1/2 = 4/(\rho+1)^2 + 4/(1-\rho)^2$$

$$1 = 8/(\rho+1)^2 + 8/(1-\rho)^2$$

$$= 8\,\frac{(\rho+1)^2 + (1-\rho)^2}{(1-\rho^2)^2}$$

$$(1-\rho)(1-\rho) = 8\left[(\rho+1)^2 + (1-\rho)^2\right]$$

$$= 8\left\{\rho^2 + 2\rho + 1 + 1 - 2\rho + \rho^2\right]$$

$$= 16\left(\rho^2 + 1\right)$$

$u = \rho^2$

$$(1-u)^2 = 16(u+1)$$

$$u^2 - 2u + 1 = 16u + 1$$

$$u^2 - 18u - 15 = 0$$

$$u = \frac{18 \pm \sqrt{324 - 60}}{2} = 9 \pm 4\sqrt{5}$$

$$\therefore \rho = \sqrt{9 + 4\sqrt{6}} \quad c \quad \checkmark$$

8. PLA vs SVM $N = 10$

better 60% of the time $\rightarrow$ c ✓

9. $N = 1000$ better 50% $\rightarrow$ c ✗

10. 3 support vectors $\rightarrow$ b ✓

LECTURE 15: KERNEL METHODS

remember → nonlinear transforms we only pay for $z^T z$ inner product computation

extend SVMs further!

- KERNEL TRICK ← lets us play in $\mathbb{R}^\infty$!!
- SOFT-MARGIN SVM ← allow some errors

we will probably use both tricks on most SVM problems

What do we need from z-space?

dealing with inner product w/ kernel trick

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \underbrace{z_n^T z_m}_{①}$$

① do we need anything other than this in z-space?

constraints: $\alpha_n \geq 0$ for $n = 1, \ldots, N$ & $\sum_{n=1}^{N} \alpha_n y_n = 0$

$g(x) = \text{sign}(w^T z + b)$ where $w = \sum_{z_n \text{ is SV}} \alpha_n y_n z_n$

($w^T z$ needs $z^T z$!)

b from $y_n(w^T z_n + b) = 1$ ← needs $z^T z$!

can we get $z^T z$ without visiting zspace or knowing what z-space is...

GENERALISED ... INNER PRODUCT

given two points x and $x' \in X$, we need $z^T z'$

(z is $f(x)$; z' is $f(x')$)

let $z^T z = K(x, x')$ ← THE KERNEL "INNER PRODUCT" of x and x'

EXAMPLE $x = x_1, x_2 \rightarrow$ 2ND ORDER Φ

$$z = \Phi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

$$K(x, x') = z^T z = 1 + x_1 x_1' + x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 + x_1 x_1' x_2 x_2'$$

THE TRICK - CAN WE COMPUTE $K(x, x')$ w/o TRANSFORMING x & x'?

Consider $K(x, x') = (1 + x^T x')^2$ ← not inner product in X

$$= (1 + x_1 x_1' + x_2 x_2')^2 \quad (1)$$

$$= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'$$

(1) THIS IS AN INNER PRODUCT

$$z = \Phi(x) = (1, x_1^2, x_2^2, \underbrace{\sqrt{2} x_1}_{c}, \sqrt{2} x_2, \sqrt{2} x_1 x_2)$$

THE POLYNOMIAL KERNEL

$\mathcal{X} = \mathbb{R}^d$ and $\Phi: \mathcal{X} \to \mathcal{Z}$ is a polynomial of order Q

"equivalent" kernel $K(x, x') = (1 + x^T x')^Q$

$= (1 + x_1 x_1' + x_2 x_2' + \ldots + x_d x_d')^Q$

↑ computationally easy

for $d = 10$ and $Q = 100$ explicitly is... big, unpleasant, etc. so don't do the transform.

ok but we have $\sqrt{}$ terms c ... but we can scale $K(x, x') = (a x^T x' + b)^Q$

WE JUST NEED $\mathcal{Z}$ TO EXIST!

if $K(x, x')$ is an inner product in <u>some</u> space $\mathcal{Z}$, this works!

$K(x, x') = \exp(-\gamma \|x - x'\|^2)$

↑ infinite-dimensional $\mathcal{Z}$ ← wat.

TAKE SIMPLE CASE

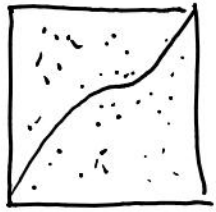
$K(x, x') = \exp(-(x - x')^2)$ $\quad (\gamma = 1)$

$= \exp(-x^2) \exp(-x'^2) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}$

split into two vectors and distribute constants and it's an inner product!

IN ACTION

slightly non-separable case



TRANSFORM X into ∞-dimensional Z. pass to quad programming and we get support vectors

in this case 9 support vectors, 100 points

$E_{out} \sim 9/91$

overkill? count support vectors :)

kernel formulation of SVM

take quadratic coefficients $[Q]$. $Q_{ij} = y_i y_j x_i^T x_j$

w/ kernel: $[Q'] = Q_{ij} \cdot y_i y_j K(x_i, x_j)$

nothing else changes

FINAL HYPOTHESIS: $g(x) = \text{sign}(w^T z + b)$ in terms of $K(-,-)$

$$w = \sum_{z_n \text{ is SV}} \alpha_n y_n z_n \Rightarrow g(x) = \text{sign}\left(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b\right)$$

where $b = y_m - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_m)$

KERNEL CHANGES MODEL :)

SO HOW DO WE CHOOSE KERNELS? WHICH KERNELS ARE VALID?

$\exists$ EXISTS FOR GIVEN $K(x, x')$ ...

1. by construction
2. math properties of kernel ← mercer's condition
3. who cares? :) ← probably don't do this but it works sometimes

DESIGN YOUR OWN KERNEL

$K(x, x')$ IS A VALID KERNEL if & only if

1. it is symmetric (dot product works :))
2. the matrix $K_{ij} = [K(x_i, x_j)]$ is positive semi-definite for any $x_1, \ldots, x_n$ (Mercer's condition)

SOFT-MARGIN SVM

TWO TYPES OF NON-SEPARABLE

- slightly
- very

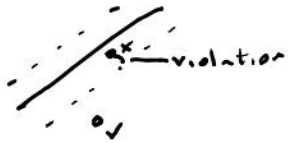
KERNELS DEAL WITH VERY NON-SEPARABLE DATA.

SOFT-MARGIN DEALS WITH SLIGHTLY "

ERROR MEASURE

margin violation (# invalid points is silly)

even if correctly classified!



QUANTIFY: $y_n(w^T x_n + b) \geq 1 - \xi_n \quad \xi_n \geq 0$

TOTAL VIOLATION = $\sum_{n=1}^{N} \xi_n$

NEW OPTIMISATION

minimise $\frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n$ ← augmented error for SVM, can see C as $1/\lambda$

① ← forces non-negative ξ_i so credit for non-violation

① subject to $y_n(w^T x_n + b) \geq 1 - \xi_n$ for $n = 1..N$

and $\xi_n \geq 0$ for $n = 1..N$

$w \in \mathbb{R}^d$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^N$

NOW WE HAVE TO GO THROUGH LAGRANGIAN AGAIN!

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1 + \xi_n) - \sum_{n=1}^{N} \beta_n \xi_n \quad (1)$$

(1) constraint on ξ ← must be ≥ 0

MINIMIZE w.r.t w, b, ξ ; MAXIMISE wrt $\alpha_n \geq 0$ & $\beta_n \geq 0$

$$\nabla_w \mathcal{L} = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \quad \text{(same as before)}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \alpha_n - \beta_n = 0 \quad \leftarrow \text{forces terms to drop out to gives us back original SVM lagrangian!!!}$$

only new constraint is that $\alpha_n \leq C$

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m \quad \text{wrt } \alpha$$

subject to $0 \leq \alpha_n \leq C$ for $n = 1, \ldots, N$

ξ', $\sum_{n=1}^{N} \alpha_n y_n = 0$

minimises $\frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n$

TYPES OF SUPPORT VECTORS

① margin support vectors $(0 < \alpha_n < C)$

$y_n(w^T x_n + b) = 1 \quad (\xi_n = 0)$

② non-margin support vectors $(\alpha_n = C)$

$y_n(w^T x_n + b) < 1 \quad (\xi_n > 0)$ ← margin violated

all margin violators are support vectors!

get C from cross-validation

can we pick kernels similarly?

TECHNICAL OBSERVATION

1. HARD MARGIN - WHAT IF DATA NON-SEPARABLE?

"primal → dual" breaks down

check if solution satisfies data

2. Z: what if there is w_0?

All goes to b and $w_0 \to 0$

EXTRA: MERCER'S CONDITION: POSITIVE SEMI-DEFINITE MATRICES HAVE ALL EIGENVALUES ≥ 0. WE'RE LOOKING FOR A SUM OF SQUARED LENGTHS FOR ALL VECTORS, SO THAT MUST BE POSITIVE. MERCER'S THEOREM STATES THAT THIS MATRIX CAN BE EXPRESSED AS A DECOMP OF EIGENFUNCTIONS - AN INNER PRODUCT

LECTURE 16: RADIAL BASIS FUNCTIONS

CAPTURES A PARTICULAR UNDERSTANING ON INPUT SPACE. ALSO A GOOD SUMMARY OF THE WHOLE COURSE!

- NEAREST NEIGHBOURS
- NEURAL NETWORKS
- KERNEL METHODS
- REGULARISATION

BASIC RBF MODEL

EACH POINT IN YOUR DATA SET $(x_i, y_i) \in D$ influences $h(x)$ based on $\|x - x_n\|$ — i.e. there is more influence on nearby data points than ones far away.

Standard form

$$h(x) = \sum_{n=1}^{N} w_n e^{(-\gamma \|x - x_n\|^2)}$$

"RADIAL" BECAUSE WE WORK IN A CIRCLE AROUND x;

$e^{\wedge}$ term is "BASIS FUNCTION"

THE LEARNING ALGORITHM

FINDING $w_1, \dots, w_N$: $h(x) = \sum_{n=1}^{N} w_n e^{(-\gamma \|x - x_n\|^2)}$

based on $\mathcal{D} = (x_1, y_1), \dots, (x_N, y_N)$

DRIVING $E_{in} = 0$: $h(x_n) = y_n$ for $n = 1, \dots, N$:

$$\sum_{m=1}^{N} w_m \exp(-\gamma \|x_n - x_m\|^2) = y_n$$

THE SOLUTION HAS N equations. AND UNKNOWNS ARE w_1 ← we have N of these. so we put in matrix form

$$\underbrace{\begin{bmatrix} \exp(-\gamma\|x_1 - x_1\|^2) & \cdots & \exp(-\gamma\|x_1 - x_N\|^2) \\ \vdots & & \vdots \\ \exp(-\gamma\|x_n - x_1\|^2) & \cdots & \exp(-\gamma\|x_n - x_N\|^2) \end{bmatrix}}_{\Phi} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

w $\quad$ y

if Φ invertible $w = \Phi^{-1} y$ "exact interpolation"

WHAT DOES γ DO? WHAT HAPPENS WHEN γ IS SMALL?
GAUSSIAN IS WIDE / FLAT

WHEN γ LARGE, GAUSSIAN IS SPIKE. POINTS HAVE LESS INFLUENCE ON EACH OTHER

THIS IS A REGRESSION MODEL. WHAT ABOUT CLASSIFICATION?

$$h(x) = \text{sign}\left(\sum_{n=1}^{N} w_n \exp(-\gamma \|x - x_n\|^2)\right)$$

use the same technique as when we used linear regression for classification - focus on signal before sign; make this signal match ± 1 target

$$s = \sum_{n=1}^{N} w_n \exp(-\gamma \|x - x_n\|^2)$$

minimise $(s - y)^2$ on $\mathcal{D}$; $y = \pm 1$

$h(x) = \text{sign}(s)$

RELATIONSHIP WITH NEAREST-NEIGHBOUR

adopt the y value of a nearby point: nearest label of closest point

tessellates decision plane into cells.

γ large might do something like this? smoothing is like applying k_{nn}.

RBF w/ K PARAMETERS

N parameters w/ N data points. but generalisation? can we pick fewer parameters? take K "centers" $K \ll N$, and focus on $\mu_1, \ldots, \mu_K$ instead of $x_1, \ldots, x_n$

$$s(x) = \sum_{k=1}^{K} w_k \exp(-\beta \|x - \mu_k\|^2)$$

but now μ_k is extremely complex - who do we manage? how do we choose centers? what about getting the weights?

choosing the centers

minimise the distance between x_n and closest center μ_k

clustering! k-means clustering algorithm

split $x_1, \ldots, x_N$ into clusters $S_1, \ldots, S_K$

minimise $\sum_{x_n \in S_k} \|x_n - \mu_k\|^2$

↑ sum over points in cluster ①

then sum over all clusters to get

$$\sum_{k=1}^{K} \sum_{①} \|x_n - \mu_k\|^2$$

good news: unsupervised learning! no reference to labels.

bad news: NP-hard! uh oh

so we need an iterative plan

LLOYD'S ALGORITHM

iteratively minimise $\sum_{k=1}^{K} \sum_{x_n \in S_k} \|x_n - \mu_k\|^2$ wrt μ_k, S_k

① $\mu_k \leftarrow \frac{1}{|S_k|} \sum_{x_n \in S_k} x_n$

now freeze μ_ks and build new clusters

② $S_k \leftarrow \{x_n : \|x_n - \mu_k\| \leq \text{all } \|x_n - \mu_\ell\|\}$

BOTH ① & ② REDUCE ERROR. WE'LL CONVERGE B/C WE'RE ON A FINITE # OF POINTS, THEREFORE FINITE CONFIGURATIONS. CONVERGENCE → LOCAL MINIMUM

TRY DIFFERENT STARTING POINTS AND SEE WHICH IS BEST.

LLOYD'S ALGORITHM IN ACTION

1. GET DATA POINTS
2. ONLY INPUTS. NO LABELS
3. INITIALISE CENTERS (CHOOSING 9 TO COMPARE W/ SVM) (RANDOM PLACEMENT)
4. ITERATE
5. AFTER CONVERGENCE WE HAVE OUR μ_ks

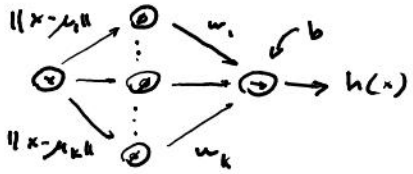
RBF CENTERS COMPARED TO SVM – THIS IS SUPERVISED VS UNSUPERVISED LEARNING

NOW WE NEED TO GO TO THE LABELS TO FIND WEIGHTS $\sum_{k=1}^{K} w_k \exp(-\gamma \|x_n - \mu_k\|^2) \approx y_n$ — N equations in K << N unknowns

$\Phi w \approx y$

if $\Phi^T \Phi$ invertible, $w = (\Phi^T \Phi)^{-1} \Phi^T y$

RELATION TO NEURAL NETWORK



YOU CAN IMAGINE AN SVM WORKS LIKE THIS - KERNEL SELECTION 1ST LAYER, THEN 2ND IS LINEAR COMBO. EVEN A KERNEL FOR NN. (TWO LAYERS)

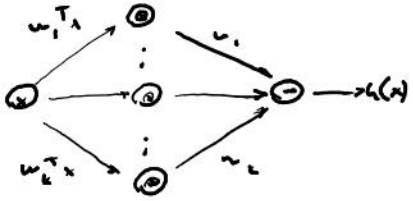
THE FEATURES ARE $\exp(-\gamma \|x-\mu_k\|^2)$

nonlinear transform depends on D

=> no longer a linear model

a bias term (b or w_0) is often added

compare w/ neural network



N.N. always contributes

RBF depends on distance

Φ is closer to linear because inputs not based on labels

CHOOSING γ

Treat γ as parameter to be learned

$$h(x) = \sum_{k=1}^{K} w_k \exp\left(-\gamma \|x - \mu_k\|^2\right)$$

CAN MINIMISE USING GRADIENT DESCENT, BUT w_k IS SIMPLE, SO MAYBE WE SPLIT w_k from γ?

ITERATIVE APPROACH: ~EM ALGORITHM

1. Fix γ, solve w_n using our pseudo-inv
2. Fix $w_1, \ldots, w_k$, minimise error
3. ITERATE :)

we can have a different γ_k for each center μ_k

.

RBF & KERNEL METHODS

SVM KERNEL IMPLEMENTS

$$\text{sign}\left(\sum_{\alpha_n > 0} \alpha_n y_n \exp\left(-\gamma \|x - x_n\|^2\right) + b\right)$$

RBF

$$\text{sign}\left(\sum_{k=1}^{k} w_k \exp\left(-\gamma \|x - \mu_k\|^2\right) + b\right)$$

SVM TRACKS TARGET FUNCTION BETTER B/C OF UNSUPERVISED LEARNING & BORDER CENTERS

RBFs AND REGULARISATION

RBF can be derived purely based on regularisation

$$\sum_{n=1}^{N} (h(x_n) - y_n)^2$$

ADD SMOOTHNESS CONSTRAINT

$$\int_{-\infty}^{\infty} \left(\frac{d^k h}{dx^k}\right)^2 dx$$

big → not smooth

$$\sum_{n=1}^{N} (h(x_n) - y_n)^2 + \lambda \sum_{k=0}^{\infty} a_k \int_{-\infty}^{\infty} \left(\frac{d^k h}{dx^k}\right)^2 dx$$

IF YOU SOLVE THIS... YOU GET RADIAL BASIS FUNCTION. YOU'RE LOOKING FOR INTERPOLATION; NOT THAT SURPRISING IT'S GAUSSIAN

EXTRAS - GO BACK TO NEURAL NETWORK COMPARISON & INTERNALISE:

LINEAR MODELS, SUPPORT VECTOR MACHINES, AND SHALLOW NEURAL NETWORKS ARE INSTANCES OF ONE TEMPLATE: A LAYER OF FEATURE DETECTORS FEEDING INTO A LINEAR COMBINATION LAYER. THE DIFFERENCE IS HOW THESE FEATURES ARE CHOSEN!

PROBLEM SET #8

1. PRIMAL VS DUAL PROBLEM

N is size of the data set, d is dimensionality of input space. The original formulation of the hard margin SVM problem w/o going through the Lagrangian dual formulation is ...

$\frac{1}{2} w^T w \; ; \; y(w^T x, w_0) \geq 1.$

$w^T = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + w_0 \quad \therefore \quad d+1$ variables $\quad d$ ✓

2. see polynomialkernels.py in repo

max E_{in} from target $= \{0, 2, \ldots, 8\}$

E_{in} target 0 : 0.1059 → a ✓

3. min E_{in} from target $= \{1, 3, \ldots, 9\}$

E_{in} target 1 : 0.0144 → a ✓

4. # SVs for target 0 : 2178

" 1 : 386

$\Delta \sim 1800 \rightarrow c$ ✓

5. RANGING C FROM $\{0.001, 0.01, 0.1, 1\}$

MAX C ACHIEVES LOWEST $E_{in} \rightarrow d$ ✓

6. WHEN $C=0.001$, # support vectors is lower for $Q=5 \rightarrow b$ ✓

7. see crossval_soft_margin.py

C selection over 100 runs:

$C:0.0001 \rightarrow 0$

$C:0.001 \rightarrow 47$ ✓ b ✓

$C:0.01 \rightarrow 27$

$C:0.1 \rightarrow 8$

$C:1 \rightarrow 18$

8. average E_{cv} for $C=0.001 \rightarrow 0.0048 \rightarrow 0.5 \rightarrow c$ ✓

9. see rbf_kernel.py

C	E_{in}	E_{out}
0.01	0.0038	0.0236
1	0.0045	0.0212
100	0.0032	0.0189
10^4	0.0026	0.0236
10^6	0.0006	"

E_{in} min when $C=10^6 \rightarrow e$ ✓

10. E_{out} min when $C=100 \rightarrow c$ ✓