Machine Learning Algorithm in grado di apprendere come risolvere uno specifico problema a partire da un set di dati

Computare area di un poligono

Rombo $D_p \cdot D_s / 2$ Dati → Algoritmo → output

Contare il n. di occorrenze di una parola in un testo

∃ Problemi con grado di incertezza → Prob. molto diff. da modellare → Classificare un imm. per stabilire quale oggetto è presente

Distinguere EMAIL → SPAM / NON SPAM → ⚠ incertezza!

Alta incertezza, non riusciamo a rappr. esaustivi per risolvere problema

Guardare contenuto EMAIL → SPAM / NON SPAM → Non vi è sempre nesso logico

DATI INPUT → Algoritmo ⇒ OUTPUT

$h_\theta(x)$

DATI INPUT, OUTPUT → Algoritmo di Learning → Programma

↓ Loss

$x^{(i)}$ → sample i

x_j → feature → colonne → Regressione, Classificazione

Train → Dtrain, Validation → Dval, test → Dtest

⚠ Problema del learning

relazione da obiettivo → dataset

Algoritmo di Learning

Un computer apprende da un'esperienza E rispetto ad un task T ad una misura di performance P, se le performance P misurate su E migliorano con l'esperienza rispetto ad un task T

$h_\theta(x)$

TASK: Problema da risolvere → Modello

↳ Funzione Parametrizzata ↳ h

valutato su

ESPERIENZA: DATI

→ misura da valutare modello sfruttando dati in input

Generalizzare

PERFORMANCE: $P\left(\{y^{(i)}\}_{i=1}^{m}, \{\hat{y}^{(i)}\}_{i=1}^{m}\right)$ → Val. Alto → ⚠ Buone Performance; Basso → ⚠ Cattive Performance

SPAM $y^{(i)}$

$h(x^{(i)}) = \hat{y}^{(i)}$

$P \in [0,1]$

$1-P \Rightarrow$ Misure di Bontà

Prob. Machine Learning → Modelli statistici che DIPENDONO dal task

Feature → Espressione di attributi — Categoriche → Polinomiali, Ordinali; Continue → numeri

Un esempio $x = (x_1, \dots, x_d)$

$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)})$ → i-esima riga del set di dati

$[\quad]^T$ → vettore riga in colonna — operiamo su questo

POSTO IN LUCE CARATT. SALIENTI DEI DATI TRASCURANDO ALTRO

Estrazione Feature

$f(x) = \bar{x}$ — Costosa a mano; Appresa

RAPPRESENTAZIONE

NON ESISTE RAPPR. universale

A partire dai dati → Estrarre caratt. importanti rispetto al task da risolvere

Tipi di Task → regressione $x \in R^d$, $y \in R$

↓

classificazione

$x \in R^d \quad y \in \{1, \dots, k\}$ k n. classi predefinito

$h_\theta(x) = y \qquad h$ funzione : $R^d \to \{1, \dots k\}$

θ → più parametrizzato

TIPI di Learning

Supervised Learning → $x \in R$, $y \in A$

Unsupervised learning $x^{(i)} \in R$

Reinforcement Learning

Provare alternative

Stimare se ALGO buono per ottenere il mio scopo

ACCURACY → CLASSIFICAZIONE → % di risposte corrette date in output dal mio algoritmo di ML

Stand-threat

Coppie di valori ideale

REGRESSIONE → ERRORE

Attributi = Features

ESPERIENZA = DATI

$x = \begin{bmatrix} x^{(m)} = x_1^{(m)} & \dots & x_d^{(m)} \\ x^{(i)} = x_1^{(i)} & \dots & x_d^{(i)} \end{bmatrix}$

colonne attributi = Feature; esempio; m feature; $x_j^{(i)}$; $x^{(i)}$

Set di dati → DATASET → Design Matrix

$x \in \mathbb{R}^{m \times d}$ → esempi × feature

Y = MATRICE $\in A^{m \times k}$ → etichette

$A = \{1, \dots, k\} \quad k = 1$

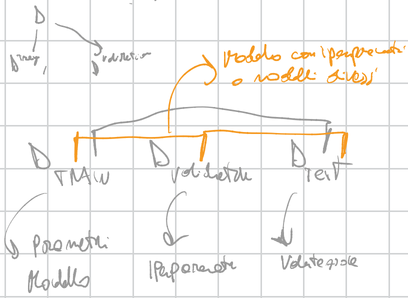
Learning / training del modello

Prende in input BANCA DATI
MODELLO (Parametri)
Misure di Performance

Output
Set di Parametri
che producono le migliori performance su D

Validazione

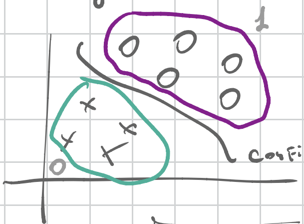
Generalizzazione



CLASSIFICAZIONE

DISCRIMINATIVA — GENERATIVA

Vista geometrica



confine di decisione → Definire stimando queste probabilità

$x^{(i)} \in \mathbb{R}^d$ $\boxed{y = \{0,1\}}$

$P(y|x)$

Regola di Bayes → $P(x|y)P(y) = P(x,y) = P(x)P(y|x)$ $P(x,y)$

Ingredienti → Classificazione

Modello

Loss function (Misura errori)

| Algoritmo di Learning

Evaluation

Semplice macchina computazionale → Neurone

1 Modello

Modello McCullogh-Pitt 1943

Mimare neurone

Unità computazionale che riceve e produce in input binari

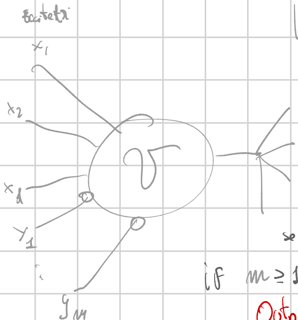
Modello return output binario

Ogni neurone è associato ad un parametro ϑ (theshold)

Output può essere connesso ad altri neuroni (anche più di uno)

Input può provenire da altri neuroni o può essere vettore binario o una computazione

Due tipi di input: Eccitatorio / Inibitorio



se inibitore acceso => output = 0 altrimenti 1

if $m \geq 1$ AND $\exists\, y_j \mid y_j = 1$ then

Output 0

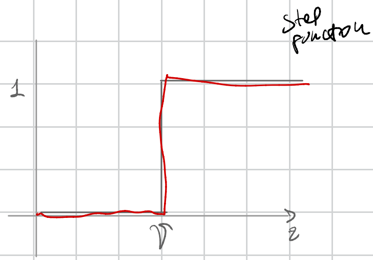
else if $\sum_{i=1}^{z} x_i \geq \vartheta$ then

Output 1

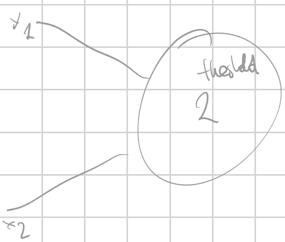
else

Output 0

Valori non binari in binari



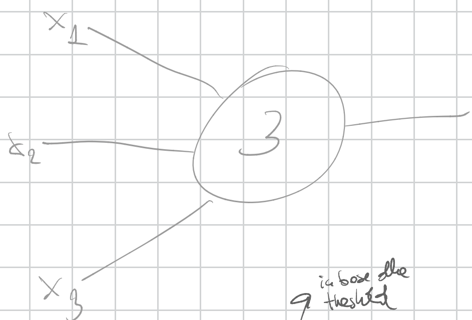
Dobb. cercare per output

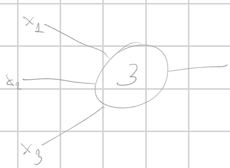


AND

$x_1 + x_2 \geq \vartheta$

x_1	x_2	output
0	0	0
0	1	0
1	0	0
1	1	1

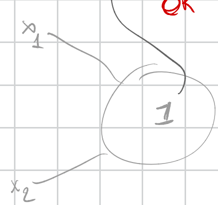




x_1	x_2	x_3	output
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
1	1	0	0
1	0	1	0
0	1	1	0
1	1	1	1

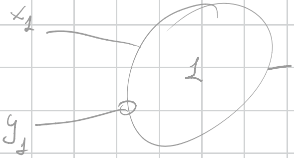
in base alla theshold

OR



Modello Parametrizzato

Funzioni lineari

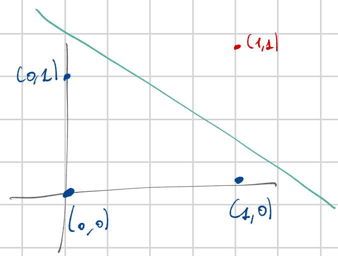


Valori soglia + inibitori = parametri

Limiti Modello

- Pensato per ricevere input binari
- Pensato solo per funzioni logiche/input binari
- Parametri fissati a mano → No procedure di learning
- * Non esistono pesi
- Tutte le funzioni sono linearmente separabili → funzione AND separiamo con classi dove possiamo mettere su asse binario

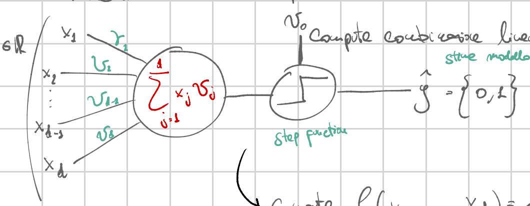
0	0	0
0	1	0
1	0	0
1	1	1



Percettrone

abbiamo settato peso ↳ lo mettiamo come input

v_0 Compute combinazione lineare del resto input



Compute $f(x_1, \dots, x_d) = \begin{cases} 1 & \text{se } \sum v_j x_j > v_0 \\ 0 & \text{altrimenti} \end{cases}$

$x_0 = 1,\ x_1^{(1)}, \dots,]$

$$[v^T x > 0] = \begin{bmatrix} v \end{bmatrix} \begin{bmatrix} X \end{bmatrix}$$

decision boundary

iperpiano del vettore del resto input

$$v_0 + x_1 v_1 + x_2 v_2 = 0$$

eq. retta

$$x_2 = \frac{v_1}{v_2} x_1 - \frac{v_0}{v_2}$$

$-0,55$

$$y = mx + q$$

Come faccio a fissare i parametri?

$\sum x_j v_j > 0$ $\quad \bar{y} = 0 \quad \hat{y} = 1$ Come posso sistemare i pesi?

↑ pixel

$x \in \mathbb{R} \rightarrow x \in \{0,1\}$ Decremento i parametri

$y \in \{0,1\}$

$y = 1 \quad \hat{y} = 0$

Incrementiamo i parametri

Errore ← Cambio!

Scelto a mano

Agg pesi

incrementare

$$v_j \leftarrow v_j + (y - \hat{y}) \, x_j$$

quantità più arbitraria in $\{0,1\}$

Formalizziamo

↗ Conoscendo valore × aggiustare pesi

Inizializzare i pesi random in $[0,1]$

Considera $x^{(i)}$ e compute $\hat{y}^{(i)}$ $\forall i$

Per le predizioni errate aggiusta i pesi

$y = 0 \quad \hat{y} = 1$ Decrementa

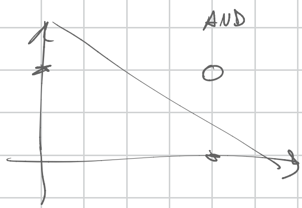
$y = 1 \quad \hat{y} = 0$ incrementa

$$v_j^{t+1} \leftarrow v_j^{t} + (y - \hat{y}) \, x_j$$

Ritorna al passo 2 fino a convergenza

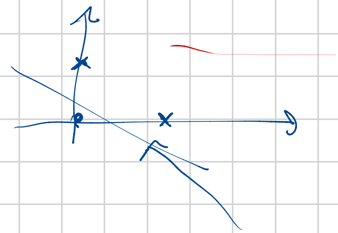
$[f_v(x) > 0]$

XOR



x_1	x_2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

XOR



$v_0 + x_1 v_1 + x_2 v_2 > 0$

$x_1 v_1 + x_2 v_2 > -v_0 \to t$

$0 v_1 + 0 v_2 < t \quad \Rightarrow t > 0$

$0 v_1 + 1 v_2 > t \quad v_2 > t > 0$

$1 v_1 + 0 v_2 > t \quad v_1 > t > 0$

$1 v_1 + 1 v_2 < t \quad v_1 + v_2 < t$

$\Big]\ v_1 > 0 \quad t > v_1 + v_2 > 2t$
$v_2 > 0$

INPUT

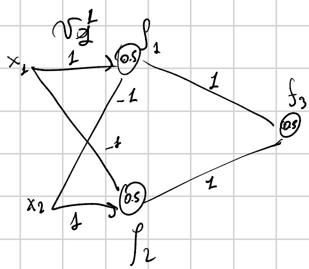
x_1	x_2	f_1	f_2	f_3
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Riesce a risolvere il problema XOR

$v_0^1 = 0.5$

$v_1^1 = 1$

$v_2^1 = -1$



Soglia $0 < 1$

$f_1(x_1 x_2) = [x_1 - x_2 > 0.5]$ — 0, 1

$f_2(x_1, x_2) = [x_2 - x_1 > 0.5]$ — 0, 1

$f_3(f_1(x_1 x_2), f_2(x_1, x_2)) = [1 f_1(x_1, x_2) + 1 f_2(x_1, x_2)]$

vettore dei pesi

$v_0^2 = 0.5$

$v_1^2 = -1$

v

Riusciamo a computare lo XOR

Modello con 3 neuroni riesce a risolvere problema XOR

Problema XOR rimane → non sappiamo come allenare

Backpropagation

Regressione

- Modello
- Loss Function → Aggiornare pesi
- Algoritmo di Learning
- Evaluation
- Test

Fase di addestramento

Fase training

TASK Regression

$f_v : \mathbb{R}^n \to \mathbb{R}^{n'}$ ⇒ vettore in input, vettore in output

parametri

valori predetti

DATI → TRAINING, VALIDATION, TEST

$\mathbb{R}^n \times \mathbb{R}^{n'}$

$\{ y^{(i)} \}_{i=1}^{m} = Y_{TEST}$

punti campioni del dataset

insieme di dati campioni

$y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ \vdots \\ y_{n'}^{(i)} \end{bmatrix}$

Mean Squared Error → Distanza Euclidea

$\| \hat{y}^{(i)} - y^{(i)} \|_2 = \sqrt{\sum_{j=1}^{n'} (\hat{y}_j^{(i)} - y_j^{(i)})^2}$

$\hat{Y}_{TEST} = \{ f_R (x^{(i)}) \}_{i=1}^{m}$

$\hat{y}^{(i)} = \begin{bmatrix} \hat{y}_1^{(i)} \\ \vdots \\ \hat{y}_{n'}^{(i)} \end{bmatrix}$

$ERRORE(Y_{TEST}, \hat{Y}_{TEST}) = \frac{1}{m} \sum_{i=1}^{m} \| \hat{y}^{(i)} - y^{(i)} \|_2$

n. di campioni predetti

media errore dei campioni

MSE

DISTANZA RADICE QUADRATA

Voglio più peso ai + ERRORI GRAVI

$f_v : \mathbb{R}^n \to \mathbb{R}^{n'}$

- $n = n' = 1$ → Simple Regression → linea → $f_v(x) = y$
- più input: $n > 1$ AND $n' = 1$ Multiple regression → piano → $f_v(x) = y$
- $n \geq 1$ AND $n' > 1$ Multivariate Regression → Object Recognition

input più predittori linear regression

vale per tutti

Modello Linear Regression

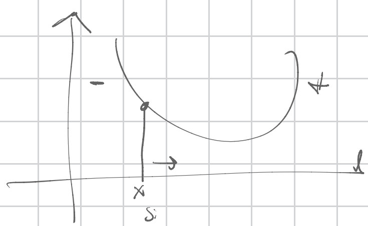
Loss Function

$f_v(x) = v_0 + x_1 v_1 + \dots + x_n v_n$

$J(v) = \frac{1}{2} \sum_{i=1}^{m} (f_v(x^{(i)}) - y^{(i)})^2$ DIFFERENZIABILE

training

$\hat{v} = \arg\min_{v} J(v)$ → quanto che minimizza l'errore



Si sposta variabile

$v_1^{(0)} =$

usando slope e regolare

$$\frac{\partial J(v)}{\partial v_j} = \sum_{i=1}^{m} (v_0 + x_1 v_1 + \dots + x_n v_n - y) x_j = \sum_{i=1}^{m} (f_v(x^{(i)}) - y^{(i)}) x_j$$

$$v_j \leftarrow v_j - \gamma \sum_{i=1}^{m} (f_v(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

learning rate dto = problem

mi garantisce minim. discesa perché

$$f_v(x) = v^T x = v_0 + v_1 x_1 + v_2 x_2 + \dots + v_d x_d \qquad x \in \mathbb{R} \quad d > 1$$

$d = 1$ $\qquad v_0 + v_1 x_1$

$\phi(x) \in \mathbb{R}^{d}$

Polinomiale

$d = 1$ $\qquad x_1 \xrightarrow{\phi} \begin{matrix} x_1 \\ x_1^2 \\ x_1^3 \\ \vdots \\ x_1^{d'} \end{matrix}$

$$f_v(\phi(x)) = v_{d'}^T x = v_{0,d'} + v_{1,d'} x_1 + v_{2,d'} x_1^2 + \dots + v_{d'} x^{d'}$$

$d > 1$ Interaction Term

$$J(v) = \frac{1}{2} \sum_{i=1}^{m} (f_v(x^{(i)}) - y^{(i)})^2$$

$\boxed{\text{Loss Function}}$

Trovare argomenti che minimizzano la funzione $J(v)$

Discesa del Gradiente

Valutazione

quanto peggio trovi questo punto

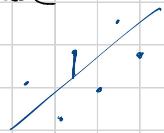
$$\text{MSE} \quad \frac{1}{m} \sum_i f_{\hat{v}} (x^{(i)}) - y^{(i)})^2 \approx J(v)$$

MAE

$$\frac{1}{m}\sum_{i=1}^{m} \left| \rho_\theta(x^{(i)}) - y^{(i)} \right|$$

∝ Semantica problema

ove da la distanza semantica più precisa



idea di distanza → MINIMIZZARE!

RMSE → RMSE = $\sqrt{MSE}$ → Come si distribuiscono

BANCA dati → SPLIT

↳ Si mantiene

Roc Curve → Casi regressione

Campioni di test ∀ coppia di test $(x^{(i)}_{test}) \xrightarrow{MSE} \varepsilon_i$ → valutiamo errore

$x^{(0)}$ ε_0 → errore

$x^{(i)}$ ε_1

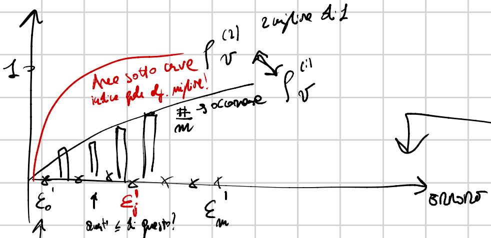
⋮

$x^{(m)}$ ε_m

ERRORI PICCOLI

↳ come faccio ad osservare?

↳ SORT → $\varepsilon'_0 \dots \varepsilon'_m$



Quanti di elementi

↳ quali di essi predicono ≤ m?

n. di elementi $x^{(i)}$ con errore $\le \varepsilon'_j$

Maggiore area sotto curva → migliore il risultato

Voglio aree piccole

es

$m = 5$

ε'_1	ε'_2	ε'_3	ε'_4	ε'_5
$\frac{2}{m}$	$\frac{3}{m}$	$\frac{3}{m}$	$\frac{4}{m}$	$\frac{5}{m} = 1$

Feature

$$\Gamma = \begin{bmatrix} x_0^{(1)} & \dots & & x_d^{(1)} \\ x_0^{(2)} & \dots & & \\ \vdots & & \dots & \\ x_0^{(m)} & & \dots & x_d^{(m)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} = Y$$

eq. normale

Teta ottimale

$$\hat{\vartheta} = \underset{\vartheta}{\mathrm{min}}\, J(\vartheta) = (\Gamma^T \Gamma)^{-1} \Gamma^T Y$$

$O(d^3)$ → d n feature per n piccoli ok, if $d >>>>>$ complessità esplode

$(d+1) \cdot m \quad m \times (d+1)$

$\Downarrow$

$(d+1) \times (d+1)$

$(d+1) \times m$

$(d+1) \times m \quad m \times 1$

$\Downarrow$

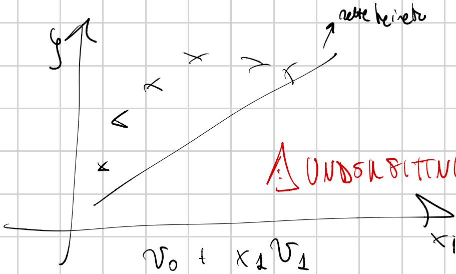
$(d+1) \times 1$

Questo deve essere invertibile

Capacità del Modello

Complessità Modello

Modelli complessi risolvono prob complessi



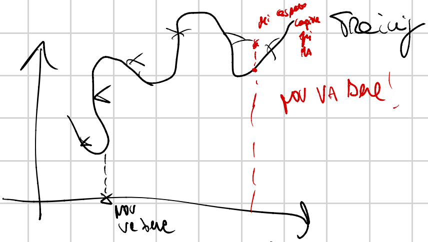
$v_0 + x_1 v_1$

Modello lineare va bene

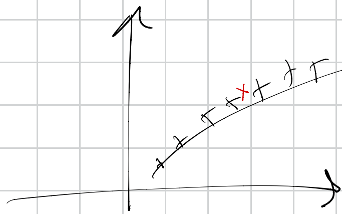
errore alto training
" " test

UNDERFITTING → Modello non capace a generalizzare su campioni nuovi

HIGH bias



$v_0 + x_1 v_1 + x_1^2 v_2 \dots$



Fa bene
training e pure del test

$v_0 + x_1 v_1 + x_1^2 v_2$

$J(v)_{TRAIN}$

$J(v)_{TEST}$

errore basso train
errore alto test

OVERFITTING
HIGH VARIANCE

cambiando di poco training
modello cambia totalmente!

errore basso train
errore basso test

Modello troppo capace → :) training
test NO!

vogliamo minimizzare questa

Regressore

→ stima errore su dati in base al modello

$$Loss\,(f_v(x), y) = J(v)$$

$$+1000\, \underset{\approx 0}{v_3} + 1000\, \underset{\approx 0}{v_4}$$

semplificando il modello!

Durante le fase di training
invece di $J(v)$
utilizziamo prior che penalizza

Vogliamo semplificare il modello

$$\underbrace{J(\theta)}_{\text{DATA TERM}} + \frac{1}{2}\lambda \underbrace{\sum_{j=1}^{k} \theta_j^2}_{}$$

→ Loss Function Regolarizzata

Regolarizzazione
previene overfitting
evita modello troppo complesso per i miei dati

λ → Iperparametro

Cerchiamo il miglior λ
1:01 importante

OK!!!

Alg. di learning → Discesa del gradiente

Model Selection → Set di dati

$J_{TRAIN-Rep}(v)$ $\quad J_{validation}(\hat{v})$

coincide con MSE $\approx J_{TEST}(\hat{v})$

TRAIN	VALIDATION	TEST
60%	20%	20%

$$f_v^{(1)}\, x^{(i)} = v_0 + x_1 v_0$$

$$f_v^{(2)}\, x^{(i)} = v_1 x_1 v_2 + x_2^2 v_0$$

$$\vdots$$

$$f_v^{k}\,(x^{(i)})$$

$\hat{v}^{(1)} = \min J_{TRAIN}(v^{(1)})$ $\qquad J_{VAL}(\hat{v}^{(1)})$

$\hat{v}^{(2)}$ $\qquad J_{VAL}(\hat{v}^{(L)})$

errore modello dopo training su validation

stimato

$\hat{v}^{(k)}$ $\qquad J_{VAL}(\hat{v}^{(k)})$

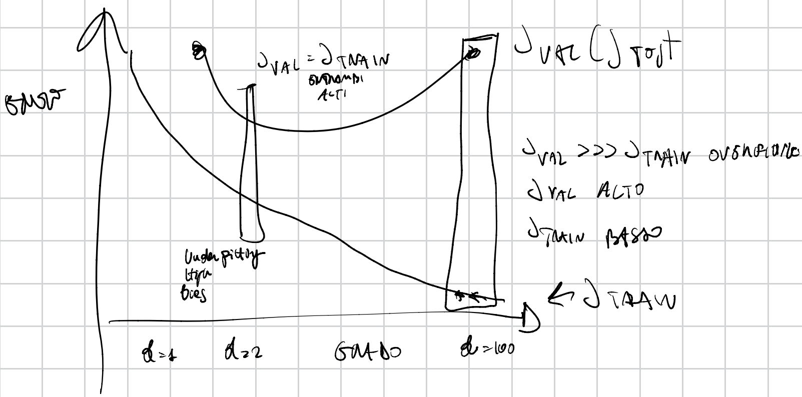
ARGMIN $J\hat{v}^{(i)}$

minimizzando funzione loss

min. errore su validation!

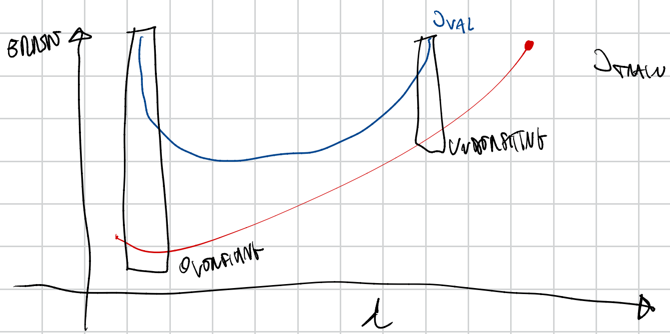
Prendiamo minimo = selezioniamo modello

Valuto modello con $J_{TEST}(\hat{v}^{(i)})$



Vediamo un po di errori e valutare

DEBUG → Plottare errori



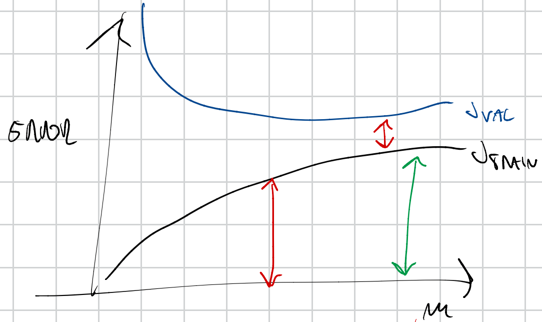
$$J_{TRAIN} + \frac{1}{2}\lambda\sum v_j^2$$

Come scelgo λ?

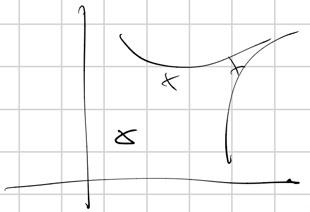
$\lambda^{(1)} = 0.01$ $\quad v^{(1)}$ $\quad J_{val}(v^{(1)})$

$\lambda^{(2)} = 0.02$ $\quad v^{(2)}$

Aumentando n. campioni base dati migliore performance?



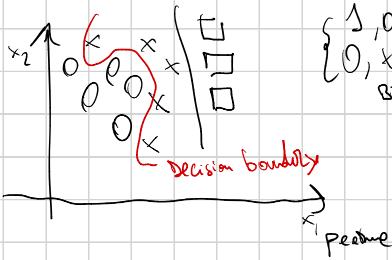
$\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$



underfitting aumentare m non ci aiuta!
overfitting aumentare m ci aiuta!

→ BASSI J^{TRAIN} ALTI su J^{VAL}

Classificazione



$\{1, 0\}$
$\{O, X\}$
Binario

Modello Parametrico
Loss Function
Alg Learning
Valutare

$x \in \mathbb{R}$ $\qquad$ $y \in \{0, ..., k-1\}$

Multiclasse
Più classificatori binari
ONE vs ALL
ONE vs ONE

$n = 10$

$d = 2$

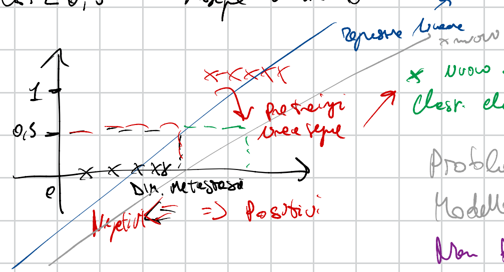
1	True	Positivo
0	Non true	Negativo

Scala 1 = dimensione pixel

class. binaria
$\{0, 1\}$
$h_{\hat{\theta}}(x) \in \mathbb{R}$
$h_{\hat{\theta}}(x) \geq 0{,}5 \rightsquigarrow$ Assegna classe 1

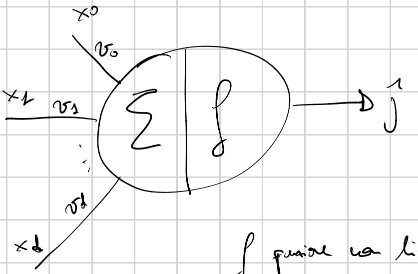
$h_{\hat{\theta}}(x) < 0{,}5 \rightsquigarrow$ Assegna classe 0

min. errore predizione



x Nuovo punto!
Class. classe 1!

Problema!
Modello $\quad 0 \leq h_{\theta}(x) \leq 1$
Non buona idea ottenere



f funzione non lineare

Modello

$$h_v(x) = \sigma(v^T x)$$

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z}\,\frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$

Sigmoide

$$v = \begin{bmatrix} v_0 \\ \vdots \\ v_d \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ \vdots \\ x_d \end{bmatrix}$$

$$h_v(x) = \sigma(v^T x) = P(y=1 \mid x, v)$$

$$P(y=1 \mid x, v) + P(y=0 \mid x, v) = 1$$

$$P(y=0 \mid x, v) = 1 - h_v(x)$$

$h_v(x) \geq 0.5 \longrightarrow$ Classe 1

$h_v(x) < 0.5 \longrightarrow$ classe 0

$$\sigma(v^T x) = \frac{1}{1+e^{-v^T x}}$$

Proprietà

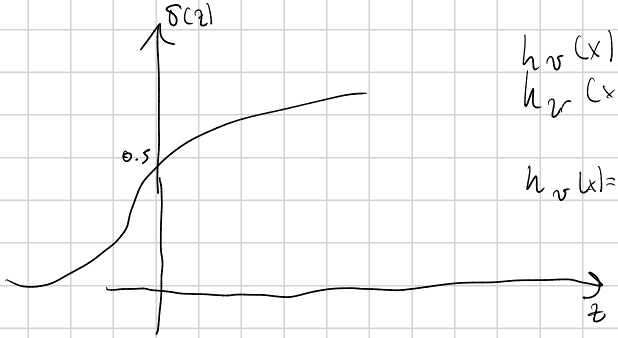
$$1 - \sigma(z) = \sigma(-z)$$

Derivata $\sigma(z) \Rightarrow \dfrac{d\sigma(z)}{dz}$

$$= \sigma(z)\cdot(1-\sigma(z))$$

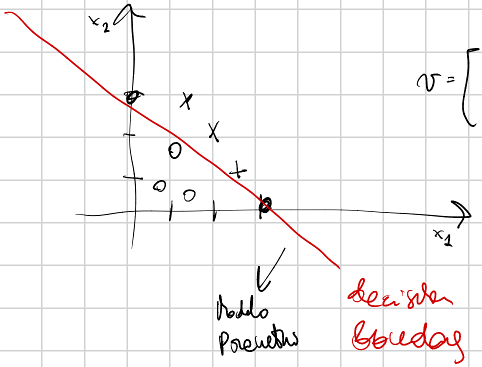
$$= \sigma(z)\,\sigma(-z)$$

$$\sigma(z) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{z}{2}\right)$$



$h_v(x) \geq 0.5 \rightarrow$ Classe 1
$h_v(x) < 0.5 \rightarrow$ Classe 0

$h_v(x) = 0.5 \iff \underline{v^T x = 0}$ decision boundary



$v = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$\sigma(x) = \frac{1}{1+e^{-(v_0x_0 + v_1x_1 + v_2x_2)}}$

$v^T(x) = -3 + x_1 + x_2$

$x_1 + x_3 = 3$ → verifica coordinate

$(0,3)$
$(3,0)$

$\hat{y} = 1 \iff -3 + x_1 + x_2 \geq 0 \iff x_1 + x_2 \geq 3$

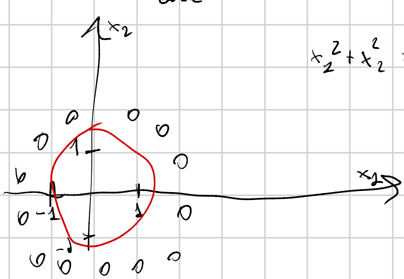
$\hat{y} = 0 \iff x_1 + x_2 < 3$

$J(v) \Rightarrow$ Funzione loss

$\phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{matrix} x_1 \\ x_2 \\ x_1 \\ x_2^2 \\ \vdots \\ x_1x_2 \end{matrix} \rightarrow \delta$

Logistic Regression → Trasformazione Polinomiale

$h_v(x) = \sigma(v^T x)$ lineare e disegna curve

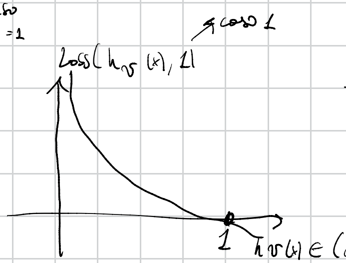


$x_2^2 + x_2^2 = 1$

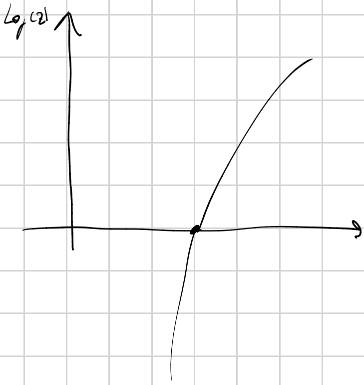
$v_0 = 1$
$v_1 = 0$
$v_2 = 0$
$v_3 = 1$
$v_4 = 1$
$v_5 = 0$

Loss Function

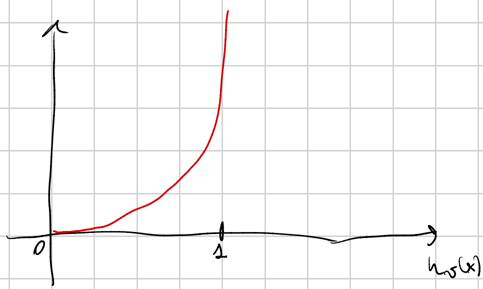
x caso $y=1$



$-\log(h_\theta(x))$

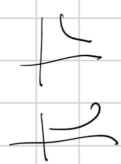


x caso $y=0$



$-\log(1-h_\theta(x))$

$$\text{Loss}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y=1 \\ -\log(1-h_\theta(x)) & y=0 \end{cases}$$



Cross Entropy Loss Binarie

$$Loss(h_v(x), y) = -y \lg(h_v(x)) - (1-y) \lg(1-h_v(x))$$

$y=1$

$y=0$

$$J(v) = \frac{1}{m}\sum_{i=1}^{m} Loss(h_v(x^{(i)}), y^{(i)})$$

Algoritmo di Learning ≅ Discesa gradiente

$$\hat{v} = \arg\min_{v} J(v)$$

$$\text{output} = h_{\hat{v}}(x) = \frac{1}{1+e^{-\hat{v}^T x}} = P(y=1 \mid x, \hat{v})$$

$$v_j \leftarrow v_j - \alpha \frac{\partial J(v)}{\partial v_j}$$

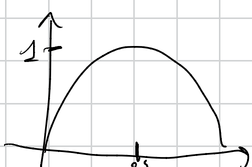
LR

$$\frac{\partial J(v)}{\partial v_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_v(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \quad \forall j = 0, \dots, d$$

$v^T x$

$\sigma(v^T x)$

$$H(X) = -\sum_{k=1}^{K} p_k \lg(p_k)$$



$P(X=1)$

$P_1 = 1$

$P_2 = 2$

= Entropia >>

stesso caso mag

$-0 \lg(0) - 1 \lg 1 = 0$

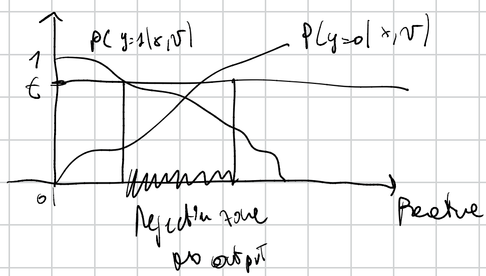
$$-y \log(\underbrace{h_\nu(x)}_{p_1}) - (1-y)\log(\underbrace{1-h_\nu(x)}_{p_2})$$

Cross Entropy Loss

$$H_p(x) = \frac{1}{m}\sum_{i=1}^{m}\left(-\sum_{k=1}^{K} p_k^{(i)} \log(q_k^{(i)})\right)$$

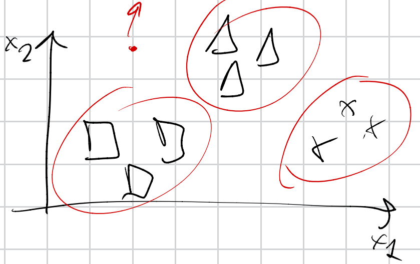
$p_k^{(i)}$: prob date dalle nostre etichette

$q_k^{(i)}$: prob imputate con il nostro classificatore



Problemi di classificazione possono essere più complessi

e come capirlo?



Modello che divide in parti uguali

Classificazione Multiclasse?

1 Δ 2 □ 3 × → classi

Suddividere problema in 3 sottoproblemi

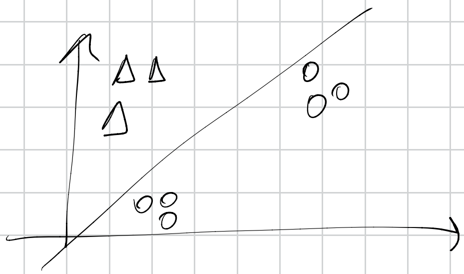
1 Δ

0 = □ ×

$h_\theta^{(1)}(x)$

class. logistico

set di parametri che determina classe 1 Δ con altre classi



$P(y = Δ | x)$

ulteriore classificazione

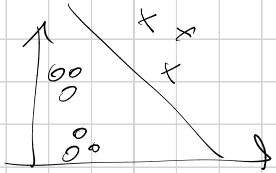
1 = ×

0 = Δ □

$h_\theta^{(3)}(x)$

× proprio

$P(y = × | x)$



Nuova ridichiarazione dato

1 = □

0 = Δ ×

$h_\theta^{(2)}(x)$

$P(y = □ | x)$

3 classificatori → Dato pattern nuovo, a quale pattern corrisponde

$$\hat{h} = \max_h h_\nu^{(k)}(x^{min})$$

$$\sum_{i=1..k \text{ classificatori}} h_\nu^{(i)} \neq 1$$

→ Appresi in maniera separata

$h_\nu^{(1)}(x^{min}) = 0.3$

$h_\nu^{(2)}(x^{min}) = 0.8$ ⇒ classe 2 → classe massima!

$h_\nu^{(3)}(x^{min}) = 0.1$

Modelli separabili in classi si può usare questa strategia

One vs ALL →

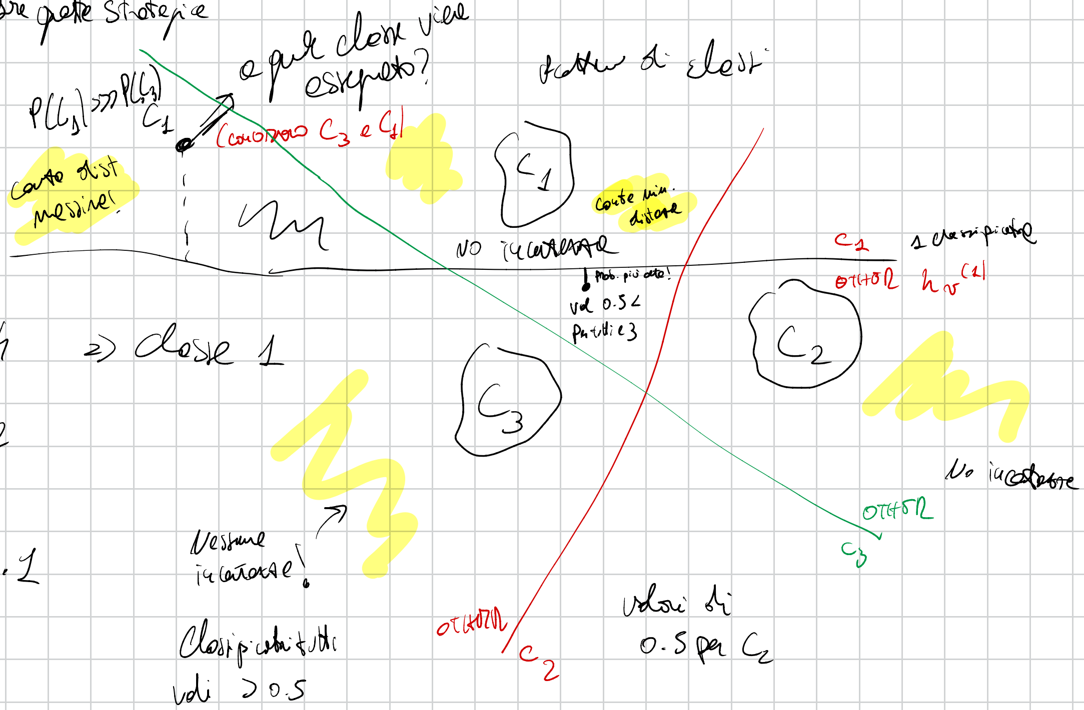
Pattern 1 vs tutti e così via

Ora supponiamo che

$h_\nu^{(1)}(x^{min}) = 0.3 / 0.4$ ⇒ classe 1

$h_\nu^{(2)}(x^{min}) = 0.8 / 0.2$

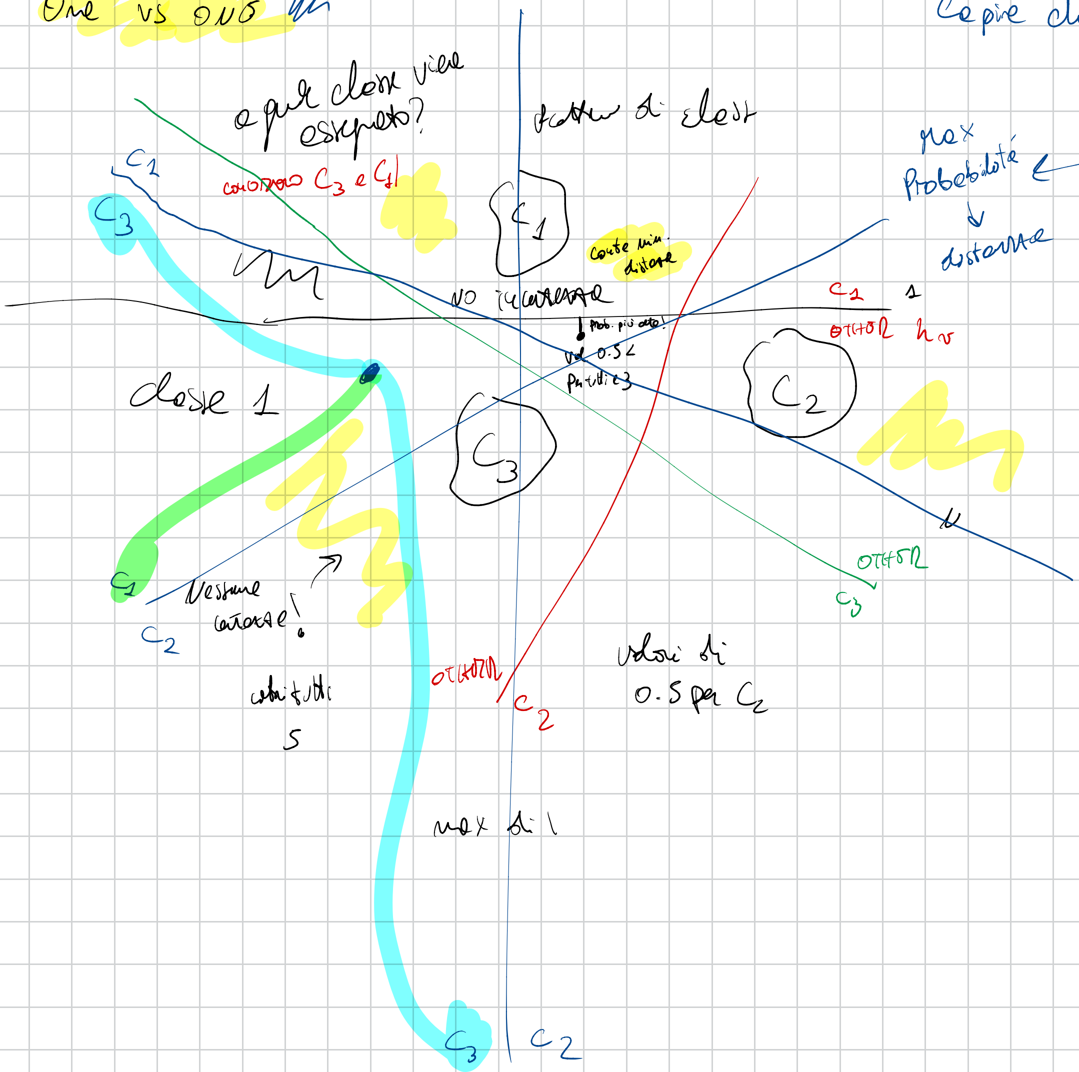
$h_\nu^{(3)}(x^{min}) = 0.1 / 0.1$



In quelle aree class. in base a max distanza di boundary

Trasp binario in multiclasse

One vs one → colore blu



Capire chi sta plottando cosa

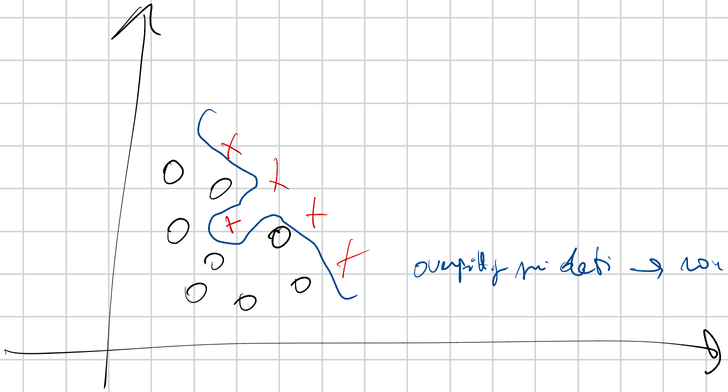
1	0	2

max
Probabilità ← 1 1 1 ?
↓
distanza

$h_v^{(1)}$ $h_v^{(2)}$ $h_v^{(3)}$

0.3 0.4 0.6

Overfitting in classificazione? Sì!



overfitting nei dati → non riusciamo a generalizzare!

Overfitting in questo caso come lo risolviamo?

Aggiungiamo la funzione il regolarizzatore
31:10

Overfitting → Regolarizzazione nella funzione loss

Generalizzazione Logistic Regression ... 33:34

Softmax

Slide

tutte sotto come una matrice

8 → 40 322

18 12 6 ↗ 16!
20 12 8
3 2 1

6 4 2
54 54 1