Università degli studi di Torino

SCUOLA DI SCIENZE DELLA NATURA

Corso di Laurea Magistrale in Informatica

Appunti del corso di Apprendimento Automatico

Autore:
Falchi Lorenzo

Anno Accademico 2023/2024

Questi appunti sono basati sulle slide del corso di Apprendimento Automatico tenuto dai professori Rosa Meo e Roberto Esposito per l'anno accademico 2023/2024. Questo documento è il risultato dell'integrazione di slide e appunti presi a lezione. Ogni capitolo corrisponde a circa una lezione, quando possibile le lezioni sono accorpate in un singolo capitolo.

Indice

1 Cenni sui prerequisiti del corso **1**
1.1 probabilità, eventi indipendenti, probabilità congiunte e condizionate 1
1.2 media, varianza, deviazione standard 1
1.3 Teorema di Bayes 1
1.4 distribuzioni di probabilità (probability distribution functions) 1
1.5 Funzione Gussiana 1
1.6 processi di Bernoulli 1
1.7 test delle ipotesi 1
1.8 intervalli di confidenza 1
1.9 funzione T-Student 1

2 Introduzione **2**
2.1 Gli ingredienti del machine learning 2
2.2 Task 3
2.3 Modelli 5
2.4 Features 6

3 Task **7**
3.1 Classificazione 7

4 Scoring and Ranking **14**
4.1 Scoring Classifier 14
4.2 Ranking 17
4.3 Stima delle probabilità 18

5 Classificatori multiclasse **20**
5.1 Estendere classificatori da binari a multiclasse 20

Capitolo 1

Cenni sui prerequisiti del corso

1.1 probabilità, eventi indipendenti, probabilità congiunte e condizionate

1.2 media, varianza, deviazione standard

1.3 Teorema di Bayes

1.4 distribuzioni di probabilità (probability distribution functions)

1.5 Funzione Gussiana

1.6 processi di Bernoulli

1.7 test delle ipotesi

1.8 intervalli di confidenza

1.9 funzione T-Student

Capitolo 2

Introduzione

2.1 Gli ingredienti del machine learning

L'obiettivo è risolvere un problema partendo da esempi di problemi già risolti. Per parlare di **machine learning** dobbiamo avere chiari 3 concetti fondamentali:

- tasks, sono una specifica di cosa vogliamo fare (classification, regression, probability estimation; clustering,...),
- model, riguardano la modalità di risoluzione del task (linear, decision trees, naive Bayes, Knn,...);
- features, sono il modo in cui sono descritti gli esempi da utilizzare per risolvere il problema(numerical, categorical, feature construction, feature selection,...).

Esempio: l'email L'obiettivo è classificare un'email come spam o come ham. Al posto di analizzare il testo, utilizziamo delle features costruite su di esso. Qunidi in partenza abbiamo le features, che sono raggruppate in vettori, e una etichetta che può essere spam o ham. Dobbiamo scrivere una funzione che prenda in input il vettore e restituisca l'etichetta corretta. **SpamAssassin** è un programma per individuare email spam. Sulla sinistra ci sono degli score che sono associati alle features; se una featutures è vera, aumenta la probabilità che l'email sia spam.

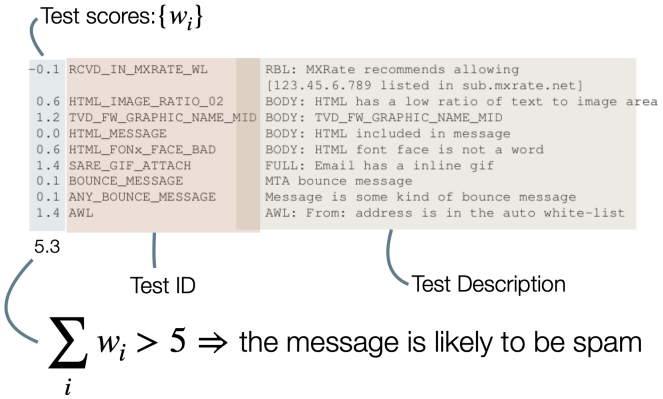


Figura 2.1: **Nota:** email **ham** è il contrario di email **spam**.

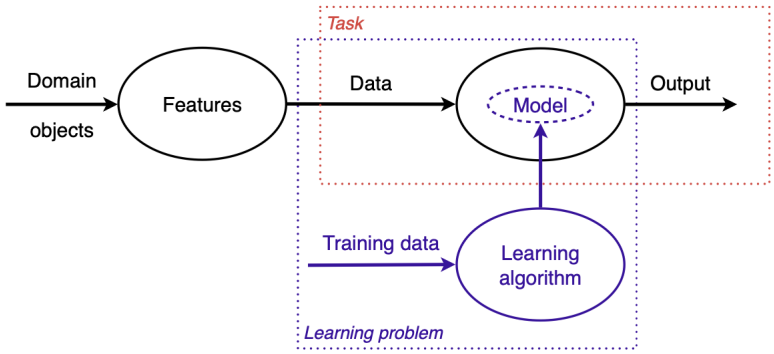


Figura 2.2: come il machine learning aiuta a risolvere un task

Machine Learning **L'apprendimento automatico** è lo studio sistematico di algoritmi e sistemi che migliorano la loro **conoscenza** o **performance**(=funzione che misura quanto il modello sta facendo bene) attraverso l'**esperienza**(=esempi, features)

Per risolvere un **task** devo **sfruttare un modello**, per risolvere un **problema di apprendimento** devo usare un qualche **algoritmo di apprendimento** che sviluppi un **modello** per risolverlo.

2.2 Task

Ci sono sostanzialmente 2 tipi di task che si affrontano con l'apprendimento automatico: **task predittivi** e **task descrittivi**. Nei predittivi si vuole predirre una qualche variabile a fronte delle descrizione degli esempi. I predittivi si disinguono a loro volta a seconda delle etichette che vogliamo predire:

- **classificazione binaria (2 etichette) o multiclasse (più etichette)**, etichetta di tipo categorico (es. spam, ham);
- **regressione**, etichetta numerica;
- **clustering**, target nascosto.

I **task descrittivi** non hanno l'etichetta fornita, l'idea è prendere i dati forniti e fornire una descrizione di essi che sia utile a chi fa l'analisi.

Overfitting. L'idea nel caso estremo è che si memorizzano troppo gli esempi e, quando si deve predirre, si tira fuori l'etichetta (se si è vista), altrimenti si va a caso. Il punto è che il modello si specializza sugli esempi forniti e non riesce a generalizzare.

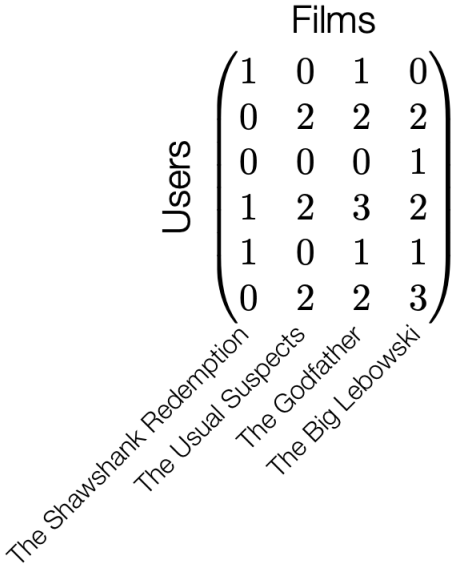


Figura 2.3: esempio di task predittivo

Le colonne rappresentano i film e le righe i giudizi delle persone ai quei film. Così è difficile tirare fuori qualche informazione utile, dobbiamo provare a generalizzare.

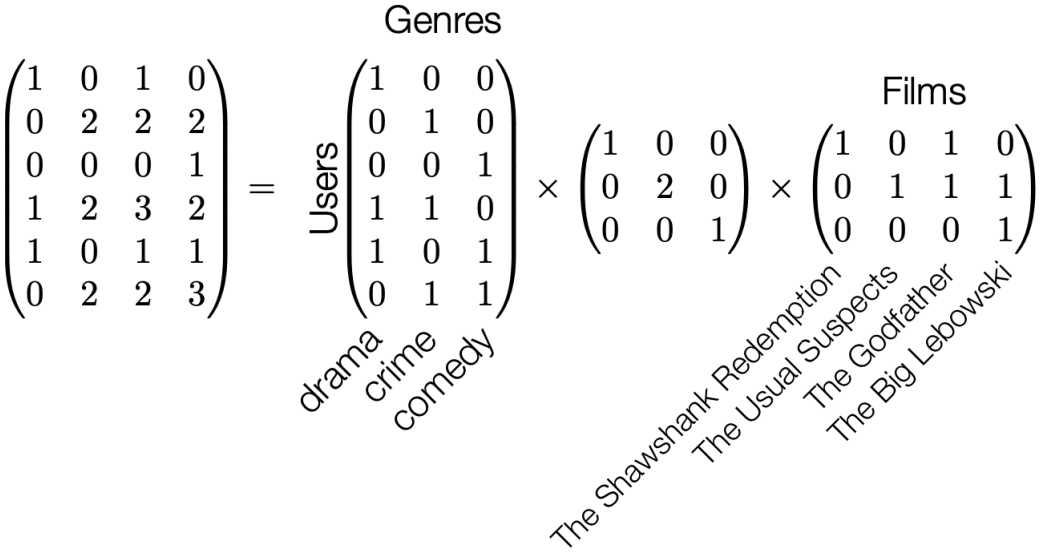


Figura 2.4

In questo modo è molto più semplice capire se ci sono relazioni tra gli utenti e i film.

2.3 Modelli

Vedremo 3 tipi di modelli:

- **geometrici**, in cui si usano le intuizioni della geometria per risolvere il problema;
- **probabilistici**, in cui il tool principale è il calcolo della probabilità e la statistica;
- **logici**, in cui il tool principale sono le espressioni logiche.

In **ambito geometrico** un algoritmo di apprendimento deve **trovare l'insieme dei pesi** che permetta alla funzione di discriminare gli esempi.

$$f_w(x) = w_1x_1 + w_2x_2 + w_3 \tag{2.1}$$

In questo esempio i pesi sono w_1, w_2, w_3 e gli algoritmi saranno algoritmi di ottimizzazione di qualche tipo.

Modelli probabilistici Un classificatore di questo tipo, usa al posto della geometria, il calcolo della probabilità di un'etichetta avendo dei dati. Sia X la descrizione del nostro esempio e Y l'etichetta che vogliamo predire, quello che vorremmo fare è trovare Y_{MAP} (cioè il massimo a posteriori), cioè la probabilità massima una volta che ho visto i dati.

$$\begin{aligned} Y_{MAP} &= arg\ max_Y P(Y|X) \\ &= arg\ max_Y \frac{P(X|Y)P(Y)}{P(X)} \\ &= arg\ max_Y P(X|Y)P(Y) \end{aligned} \tag{2.2}$$

Nota: $P(Y|X)$ è la probabilità di avere Y dato X.

Il denominatore non è importante perchè siamo interessati alla Y che massimizza la probabilità.

Spesso $P(Y)$ non è nota quindi si usa la formula

$$Y_{ML} = arg\ max_Y P(X|Y). \tag{2.3}$$

Man mano che cresce il numero di valori da considerare, la tabella che viene fuori da questa formula è enorme (per 20 valori sarebbero 2^{20} righe). In questo caso si usa uno degli algoritmi più semplici che si utilizzano con i metodi probabilistici è l'algoritmo di **Naive Bayes** che consiste nel fare un'assunzione forte su come sono fatte le probabilità (esempio assumere che x_1 e x_2 siano indipendenti tra di loro, quindi non serve calcolare l'intera tabella ma solo una per x_1 e una per x_2).

Modelli logici. Sono quei modelli in cui lo strumento che si utilizza per modellare i dati e i risultati è la **logica**. Ce ne sono molti tipi, un esempio è un classificatore logico che ha una serie di regole della forma se x allora y.

- se l'email contiene la parola "viagra" stima gli odd di spam come 4:1 (spam e 4 volte più probabile rispetto ad ham);
- se l'email contiene la parola "blue pill" stima gli odd di spam come 3:1;
- altrimenti stima gli odd come 1:6.

Quindi quando si riceve un'email si applicano una dopo l'altra queste regole e la prima che matcha da la stima degli odd da cui poi si decide se dare spam o ham come etichetta corretta.

2.4 Features

Rappresentano come descriviamo i dati e hanno importanza fondamentale per arrivare ad un buon risultato. Facciamo un esempio

Supponiamo di voler approssimare $y = \cos \pi x$ nell'intervallo $-1 \leq x \leq 1$. Un'approssimazione lineare non sarebbe molto utile in questo caso, poiché il miglior adattamento sarebbe $y = 0$. Tuttavia, se suddividiamo l'asse x in due intervalli, $-1 \leq x < 0$ e $0 \leq x \leq 1$, possiamo trovare approssimazioni lineari ragionevoli in ciascun intervallo. Possiamo ottenere ciò utilizzando x sia come caratteristica di divisione che come variabile di regressione.

Un altro esempio: si vede che l'andamento è molto frastagliato.

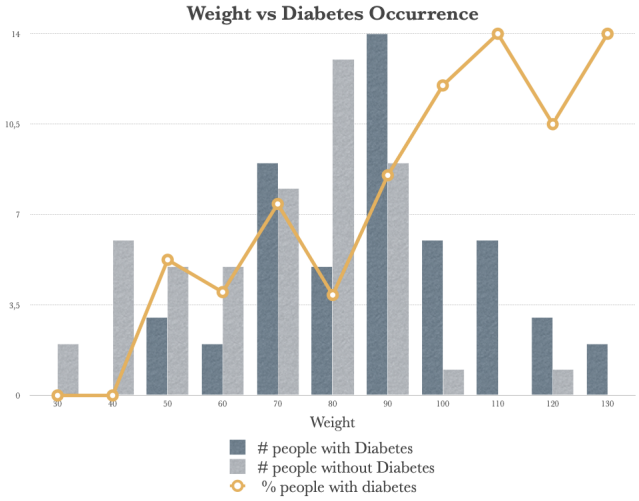


Figura 2.5: Caption

Se al posto di aggregare per 10 chili lo facciamo per 20 chili, si riesce a trattare molto meglio.

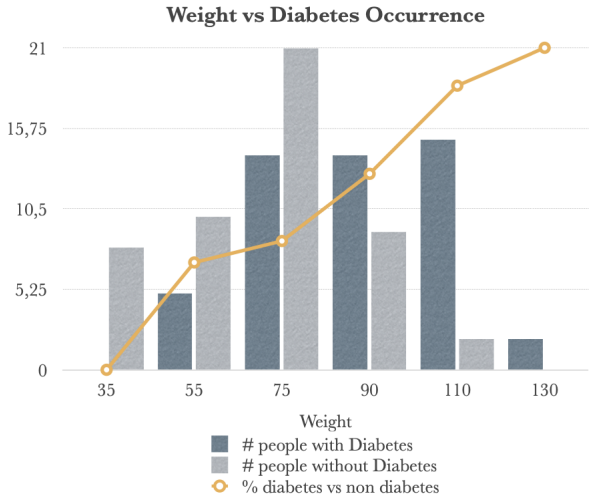


Figura 2.6: Caption

Quindi anche solo lavorando sulle features, si riesce a semplificare di molto il problema.

Capitolo 3

Task

Uno scenario di machine learning predittivo è quello in cui, dato un set di dati etichettati, viene chiesto a un algoritmo di apprendimento di costruire un modello in grado di fare previsioni su "alcune proprietà" di nuovi esempi (mai visti in precedenza).

A seconda del tipo di etichette associate ai dati e di quale "**proprietà**" l'algoritmo deve prevedere, possiamo distinguere tra i seguenti compiti: **classificazione**, **scoring** e **ranking**, **stima della probabilità** e **regressione**.

3.1 Classificazione

Un **classificatore** fa una mappatura di questo tipo:

$$\begin{aligned} \hat{c} &: X \to C \\ C &= C_1, C_2, \ldots, C_K \end{aligned} \tag{3.1}$$

X è lo **spazio degli esempi** e C è lo **spazio delle classi**. Il **cappello** sopra C indica che è un'approssimazione del concetto reale. Costruire un classificatore implica costruire la funzione $\hat{c}$ in modo tale che corrisponda a c il più fedelmente possibile (e non solo sul training set, ma idealmente sull'intero spazio delle istanze X).

Un esempio è la coppia:

$$(x, c(x)) \in X \times C \tag{3.2}$$

dove x è un' **istanza** e $c(x)$ è la classe reale a cui appartiene l'istanza.

Classificazione binaria. E' il caso particolare in cui l'insieme delle classi C contiene solo le classi 0 e 1 (o true e false). Dalla classificazione binaria si può passare a quella multiclasse senza cambiare algoritmo.

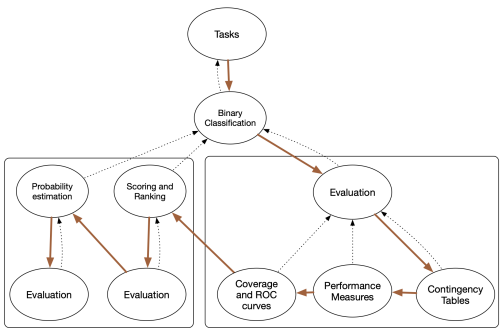
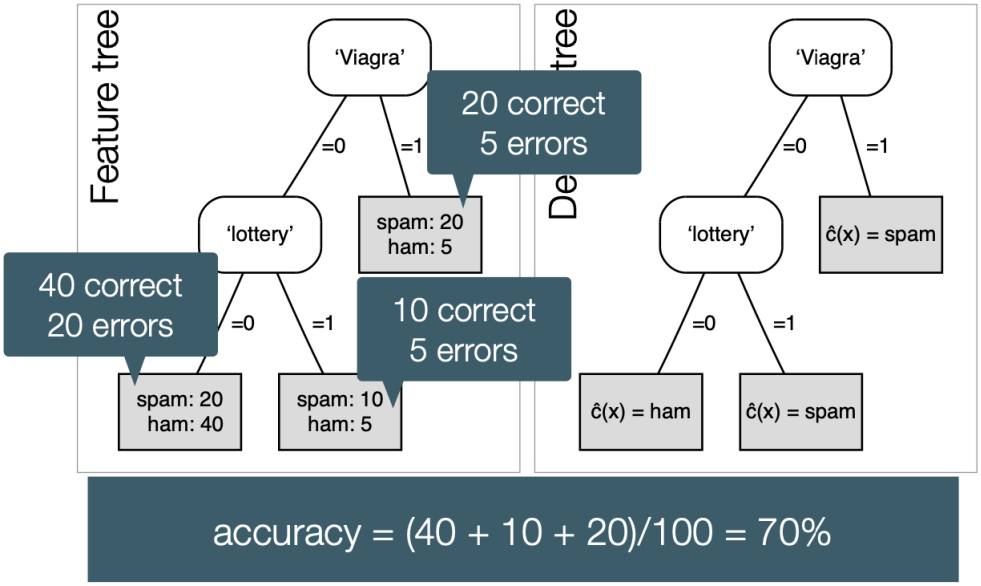


Figura 3.1: Binary classification topics

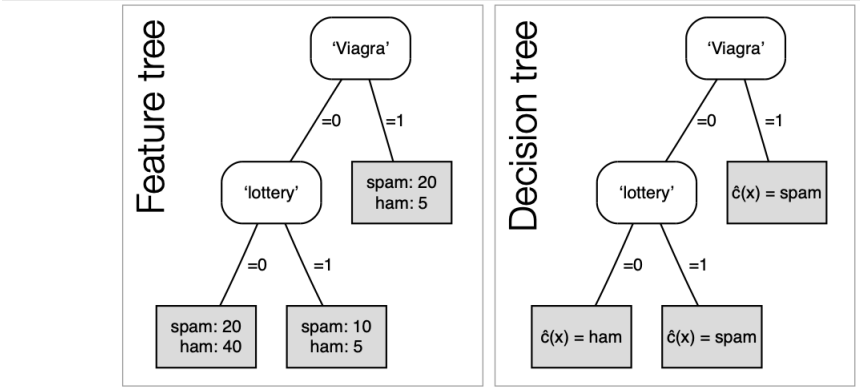
Valutazione Introduciamo gli **alberi di decisione**. L'esempio è il solito sull'etichettatura dell'email. L'idea è che ogni **nodo** corrisponde ad una **feature** ('viagra' e 'lottery' in questo caso) e se quella feature è presente seguo un ramo, altrimenti seguo l'altro. Arrivato al fondo ottengo l'etichetta.

Gli **alberi di feature** sono analoghi ma nelle foglie ho il numero degli esempi del training set che sono finiti in quella foglia, suddivisi nelle varie classi.



Guardando i numeri e le etichette si calcola l'**accuratezza**, che è una delle misure che si usano per valutare il modello. Non è l'unica però, infatti nasconde molte informazioni.

Nella **tavola di contingenza** (nel caso binario una tavola $2x2$), le colonne corrispondono a ciò che abbiamo predetto e le righe alle etichette che ci sono state date. Nelle interesezioni ci sono gli esempi che sono stati predetti in un certo modo e che hanno una certa etichetta.



	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

Figura 3.2: Tavola di contingenza

Introduciamo un po' di notazione:

- **TP**, **true positive**, quelli che ho predetto essere positivi e che sono effettivamente positivi;
- **FP**, **false positive**, quelli che ho predetto essere positivi e che sono però negativi;
- **FN**, **false negative**, quelli che ho predetto essere negativi e che sono però positivi;
- **TN**, **true negative**, quelli che ho predetto essere negativi e che sono effettivamente negativi;

$$\begin{aligned} TP + FN &= POSITIVE \\ FP + TN &= NEGATIVE \end{aligned} \tag{3.3}$$

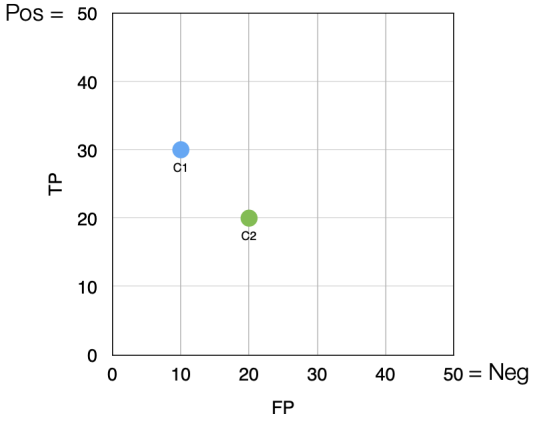
Measure	*Definition*	*Equal to*	*Estimates*
number of positives	$Pos = \sum_{x \in Te} I[c(x) = \oplus]$		
number of negatives	$Neg = \sum_{x \in Te} I[c(x) = \ominus]$	$\lvert Te \rvert - Pos$	
number of true positives	$TP = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]$		
number of true negatives	$TN = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]$		
number of false positives	$FP = \sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]$	$Neg - TN$	
number of false negatives	$FN = \sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]$	$Pos - TP$	
proportion of positives	$pos = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \oplus]$	$Pos/\lvert Te \rvert$	$P(c(x) = \oplus)$
proportion of negatives	$neg = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \ominus]$	$1 - pos$	$P(c(x) = \ominus)$
class ratio	$clr = pos/neg$	Pos/Neg	
(*) accuracy	$acc = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$		$P(\hat{c}(x) = c(x))$
(*) error rate	$err = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) \neq c(x)]$	$1 - acc$	$P(\hat{c}(x) \neq c(x))$

Figura 3.3: Misurazioni di performance

Coverage plot. Il **coverage plot** è un modo per mettere in un grafico le informazioni che ci sono nella tavola di contingenza. I FP sono sull'asse delle x, i TP sull'asse delle y e mettiamo un punto sul grafico in corrispondenza del punto considerato della tavola di contingenza.

	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

	Predicted ⊕	Predicted ⊖	
Actual ⊕	20	30	50
Actual ⊖	20	30	50
	40	60	100



Sembrerebbero mancare gli esempi che non sono stati predetti come positivi, ma questo non è un problema. Possiamo infatti calcolare i FN come $POS - TP$ e TN come $NEG - FP$. Quindi in verità nel grafico è rappresentata tutta l'informazione della tavola di contingenza.

Per cosa lo utilizziamo? Lo utilizziamo capire quale classificatore è migliore. In questo caso sembra essere c_1 poichè ha un numeri di TP più alto ma anche un numero di TN più alto.

Nel grafico ci sono 2 zone chiamate **ROC heaven** e **ROC hell**. **ROC heaven** è la zona del grafico che corrisponde a dove vorrei che fossero i miei classificatori (in alto a sinistra), al contrario **ROC hell** è una zona che descrive un classificatore che sbaglia sempre. In realtà però il punto peggiore è un altro poichè se un classificatore è nel ROC hell basta prendere il contrario delle etichette che fornisce. Il posto peggiore è la **diagonale principale**, li ci sono i classificatore che sostanzialmente tirano a caso.

Due classificatori sono uguali (cioè hanno la stessa accuratezza) se hanno la stessa distanza dall'angolo in alto a sinistra e si dice che sono su una **isometria di accuratezza**.

	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

	Predicted ⊕	Predicted ⊖	
Actual ⊕	20	30	50
Actual ⊖	20	30	50
	40	60	100

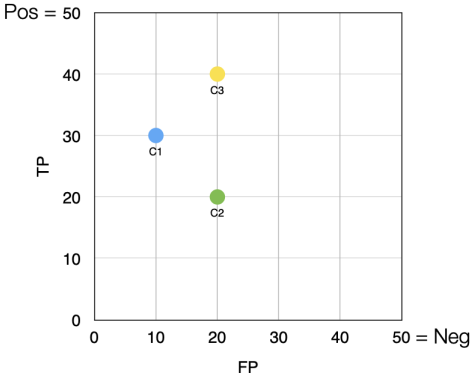


Figura 3.4: Due classificatore sulla isometria di accuratezza

Proprietà del coverage plot. Formalizziamo il concetto introdotto prima: i classificatori con la stessa **accuratezza** si trovano su una linea con **pendenza 1**. L'accuratezza è data da

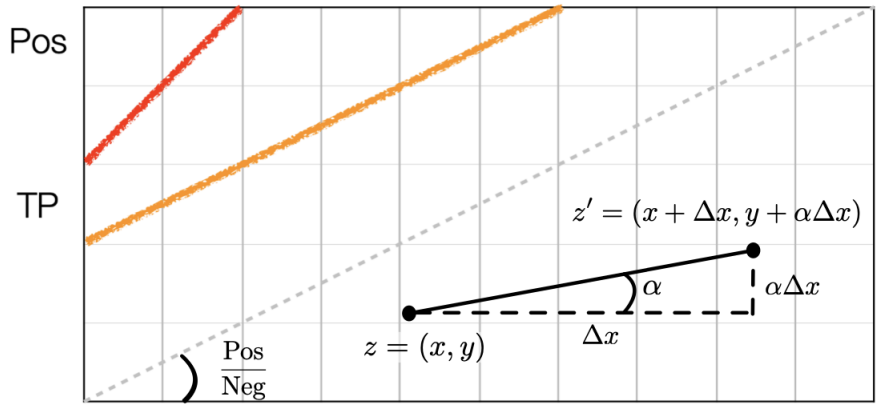
$$ACC = \frac{TP + TN}{TOT}. \tag{3.4}$$

Notiamo che se ci spostiamo a destra e in alto di uno:

$$\begin{aligned} ACC &= \frac{(TP+1)+(TN-1)}{TOT} \\ ACC &= \frac{TP+TN}{TOT}. \end{aligned} \tag{3.5}$$

Tutti i classificatori con la stessa **average recall**, sono sulla stessa linea che è **parallela alla diagonale principale**. L'average recall è la media tra la positive recall e la negative recall che sono così definite:

$$\begin{aligned} POS \cdot REC &= \frac{TP}{POS} \\ NEG \cdot REC &= \frac{TN}{POS} \\ avg \cdot recall &= \frac{(pos \cdot reg + neg \cdot rec)}{2} \end{aligned} \tag{3.6}$$



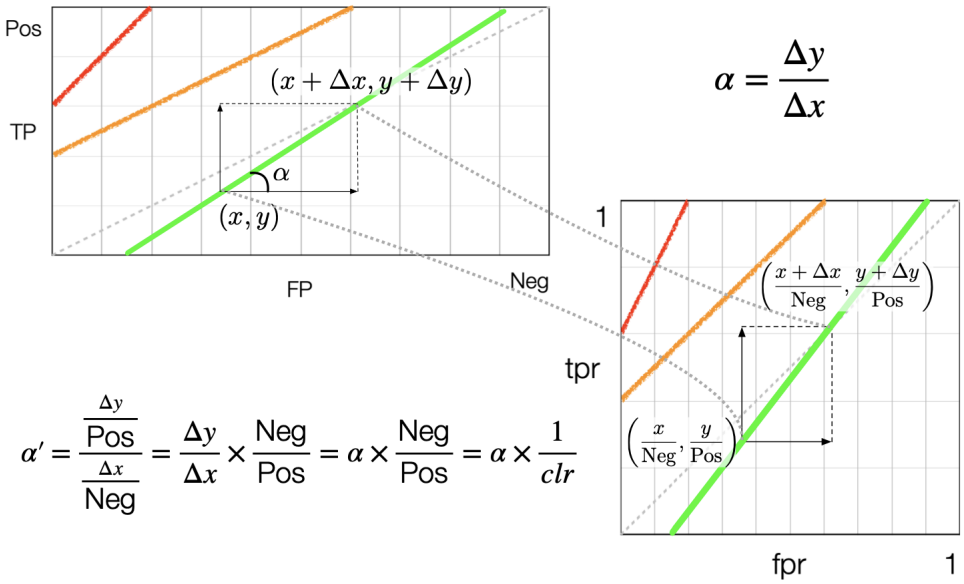
Vogliamo calcolare la pendenza della retta su cui giacciono due esempi che hanno la stessa avg recall.

$$\begin{aligned} avgrec(z) &= \left(\frac{y}{Pos} + \frac{Neg - x}{Neg}\right)/2 \\ avgrec(z) &= \left(\frac{y + \alpha\Delta x}{Pos} + \frac{Neg - x - \Delta x}{Neg}\right)/2 = \\ avgrec(z) &= \left(\frac{y}{Pos} + \alpha\frac{\Delta x}{Pos} + \frac{Neg - x}{Neg} - \frac{\Delta x}{Neg}\right)/2 \end{aligned} \tag{3.7}$$

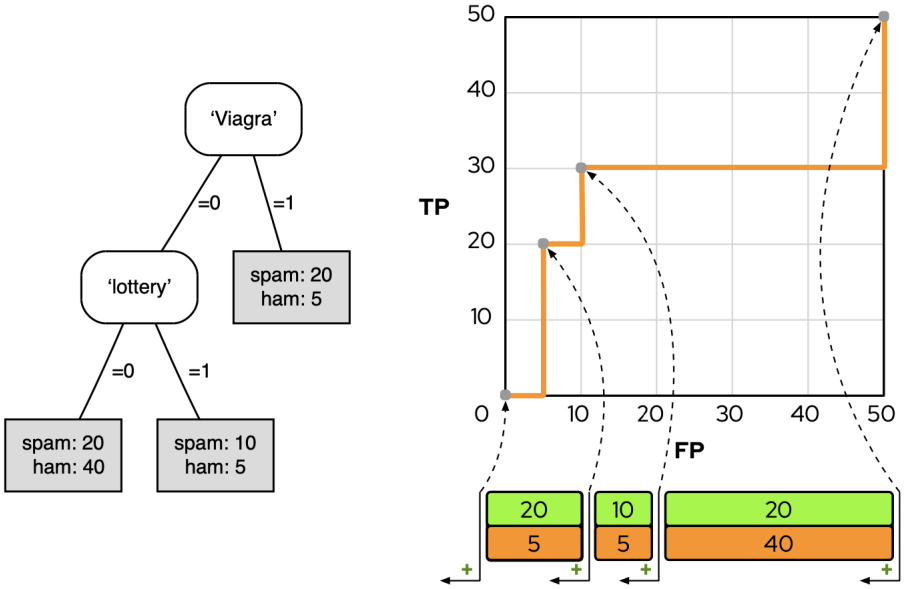
Quindi z e z' hanno la stessa avg-rec **se e solo se** $\alpha\frac{Pos}{Neg}$.

Roc Plots. L'unica differenza con i coverage plot è che normalizziamo gli assi ad dividendo le y per Neg e le x per Pos. Il vantaggio è che possiamo confrontare classificatori che sono stati addestrati su dataset di dimensioni diverse.

Vediamo le differenza con il coverage plot.



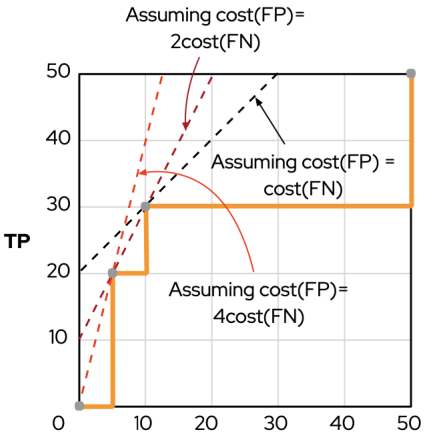
Quindi l'**accuratezza** cambia, i classificatori che stanno su rette con pendenza $\alpha = \frac{Neg}{Pos}$ hanno la stessa accuratezza. Punti che stanno su rette con pendenza 1 hanno la stessa **average recall**.



Da un singolo decision tree derivano molti classificatori. Scegliamo una **treshold** e decidiamo che tutto ciò che è alla sua sinistra è considerato negativo e tutto cioò che è alla sua destra è considerato positivo. Nel momento in cui scegliamo una nuova treshold stiamo creando un nuovo classificatore. Per esempio, se scegliamo come treshold 10 sull'asse delle x, la prima foglia è considerata positiva, le altre no (arancioni=negativi, verdi=positivi). I falsi positivi saranno 5 e i veri negativi saranno 20.

In verità, a **seconda di come assegnamo i costi ai FP e FN, la risposta alla domanda "qual è il classificatore migliore?", cambia**. Assumendo un costo $FP = FN$, allora il punto (10,30) è il migliore. Ma se scelgo un costo $FP = 2FN$ allora (10,30) e (5,20) sono sulla stessa isometrica e posso scegliere indifferentemente uno dei due. Se il costo $FP = 4FN$, il punto che sovrasta tutti sarebbe (5,20).

If $\text{cost}(FP) = C \times \text{cost}(FN)$,
i.e., if $C = \dfrac{\text{cost}(FP)}{\text{cost}(FN)}$,
then **points on lines with slope $\alpha = C$ have the same accuracy**



Quindi se C è il rapporto tra i costi, **i punti sulle rette con pendenza α hanno la stessa accuratezza**.

Ma perchè questo è vero? Innanzitutto dobbiamo ridefinire la misura di accuratezza.

$$\begin{aligned} x &= FP \\ y &= TP \\ acc = \frac{TP + TN}{TOT} &= \frac{TP + TN}{Pos + Neg} \end{aligned} \tag{3.8}$$

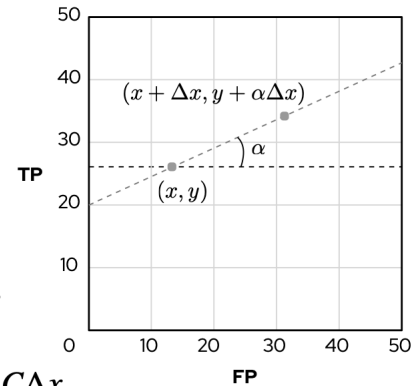
Ricoridamo che, dalla tavola di contingenza $TN = Neg - FP$, quindi

$$acc = \frac{TP + Neg - FP}{Pos + Neg} \tag{3.9}$$

Ma cosa vuol dire che il **costo dei positivi è C volte il costo dei negativi?** É come immaginare di avere un nuovo dataset in cui tutti gli esempi negativi sono replicati 4 volte. In questo modo, ogni volta che faccio un errore su un negativo, sto facendo 4 errori. Da qui:

$$\text{acc}_C(x, y) = \frac{y + C \times \text{Neg} - Cx}{Pos + C \times Neg}$$

$$\begin{aligned} &\text{acc}_C(x + \Delta x, y + \alpha\Delta x) = \\ &= \frac{y + \alpha\Delta x + C \times \text{Neg} - C(x + \Delta x)}{Pos + C \times Neg} \\ &= \frac{y + C \times \text{Neg} - Cx}{Pos + C \times Neg} + \frac{\alpha\Delta x - C\Delta x}{Pos + C \times \text{Neg}} \end{aligned}$$



$$\text{if } \alpha = C, \text{ then: } \frac{\alpha\Delta x - C\Delta x}{Pos + C \times \text{Neg}} = \frac{C\Delta x - C\Delta x}{Pos + C \times \text{Neg}} = 0$$

Nei **ROC plot**, per ottenere la stessa isometria, moltiplico per $\frac{1}{clr}$ dove $clr = \frac{Pos}{Neg}$.

Capitolo 4

Scoring and Ranking

4.1 Scoring Classifier

Cambiamo ora tipo di classificatore: prima ne usavamo uno binario a cui davamo un esempio e restituiva 0 o 1, ora invece prendiamo un classificatore che prende un esempio X restituisce un vettore:

$$\hat{s} : X \to \mathbb{R}^k \tag{4.1}$$

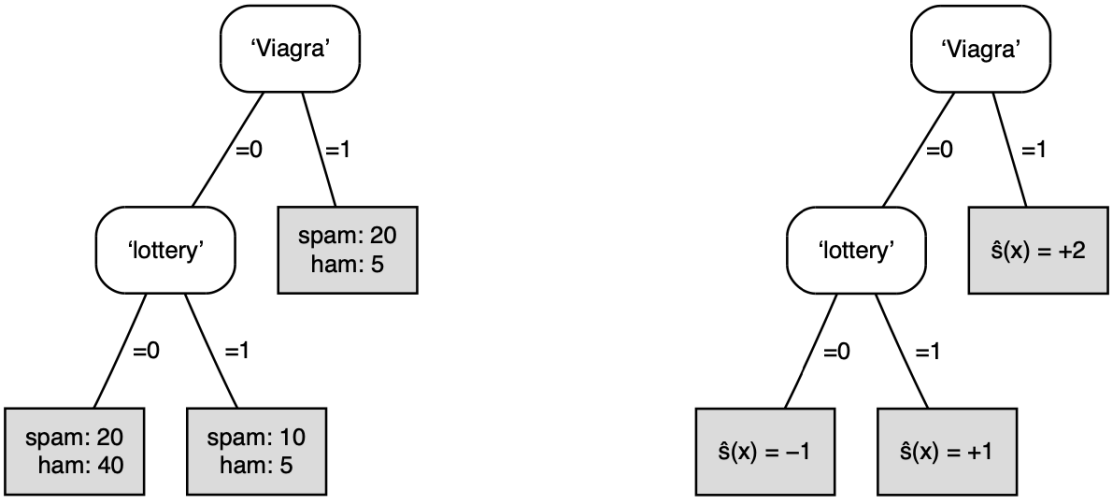
con il vettore fatto così:

$$\hat{s} = (\hat{s_1}(x), \ldots, \hat{s_k}(x)). \tag{4.2}$$

Più uno score è alto, più la classe associata a quello score, è considerata affidabile. Se ci sono solo 2 classi, gli score negativi saranno dedicati alla classe c_1 e quelli positivi alla classe c_2.

Il training set continua ad essere un esempio con una classe associata, l'output è cambiato poichè restituisce, appunto, uno score.

Scoring Tree In questi alberi, alle foglie è associato uno score invece di una classe. Per calcolare gli score si fa il logaritmo del rapporto tra, in questo caso, spam e ham.



Margini. Un **margine** è la combinazione degli score con la vera etichetta associata al modello.

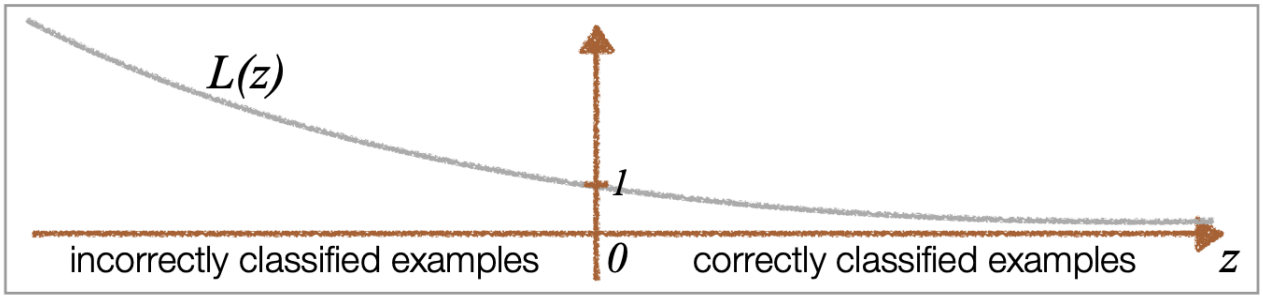
$$z(x) = c(x)\hat{s}(x) = \begin{cases} +|\hat{s}(x)| \text{ se } \hat{s} \text{ è corretto} \\ -|\hat{s}(x)| \text{ altrimenti} \end{cases} \tag{4.3}$$

$c(x)$ è la classe vera che c'è nel dataset, $\hat{s}(x)$ è lo score assegnato. Per semplicità consideriamo comunque la classificazione binaria; l'idea è che il margine sia positivo quando la classificazione è corretta e negativo quando non lo è.

Loss function. Una **loss function** è una funzione che misura quanto male sta facendo il modello. Spesso i problemi di apprendimento si definiscono in funzione delle loss function, come **problemi di minimizzazione della funzione**.

La funzione è definita come $L : \mathbb{R} \to [0, \infty)$ che mappa il margine $z(x)$ di un esempio ad un certo valore di loss $L(z(x))$.

Assumiamo che nel caso in cui il margine sia zero assumeremo che la loss sia $L(0) = 1$. Quindi la loss sarà $L(z) \geq 1$ se $z < 0$ e $0 \leq L(z) < 1$ se $z > 0$.



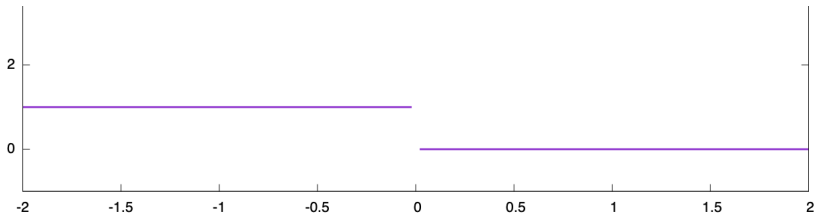
Ci sono diversi tipi di loss function:

- **0-1 loss**, la più semplice, indica che la loss è 1 se stiamo sbagliando a classificare l'esempio, 0 altrimenti

$$\begin{aligned} L_{01}(z) &= 1 \text{ if } z \leq 0 \\ L_{01}(z) &= 0 \text{ if } z > 0 \end{aligned} \tag{4.4}$$

  sommando la loss di tutti gli esempi, vediamo gli esempi che stiamo classificando male.

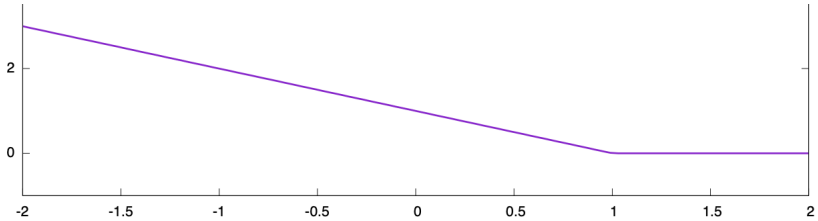
  Notiamo che la funzione è discontinua e la sua derivata è 0 sia a sinistra che a destra,



  il che vuol dire che se riesco a classificare correttamente gli esempi, non ho nessun altra informazione da utilizzare per migliorare ulteriormente il modello.

- **Hinge loss**, risolve parzialmente il problema della derivata,

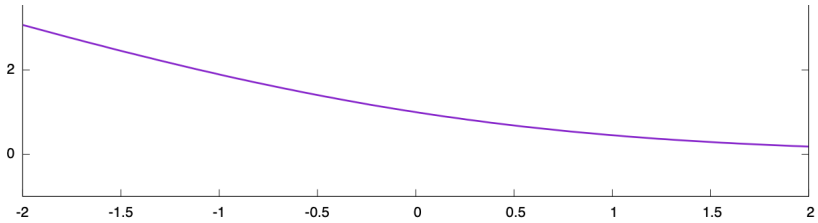
$$\begin{aligned} L_h(z) &= (1 - z) \text{ if } z \leq 1 \\ L_h(z) &= 0 \text{ if } z > 1 \end{aligned} \tag{4.5}$$



  ci permette di migliorare la classificazione anche quando siamo tra 0 e 1;

- **logistic loss**,

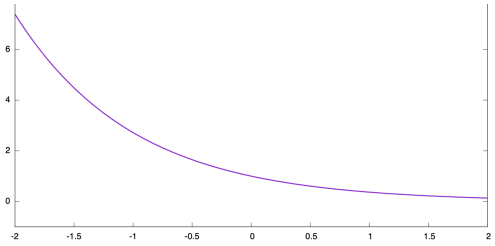
$$L_{log}(z) = \log_2(1 + (\exp(-z))) \tag{4.6}$$



è un'approssimazione della precedente, completamente continua;

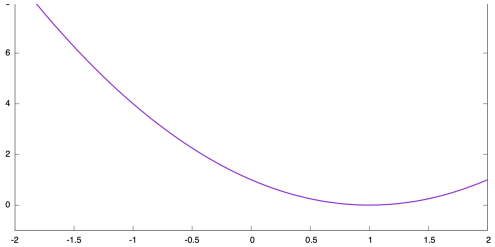
- **exponential loss**,

$$L_{exp}(z) = \exp(-z) \tag{4.7}$$

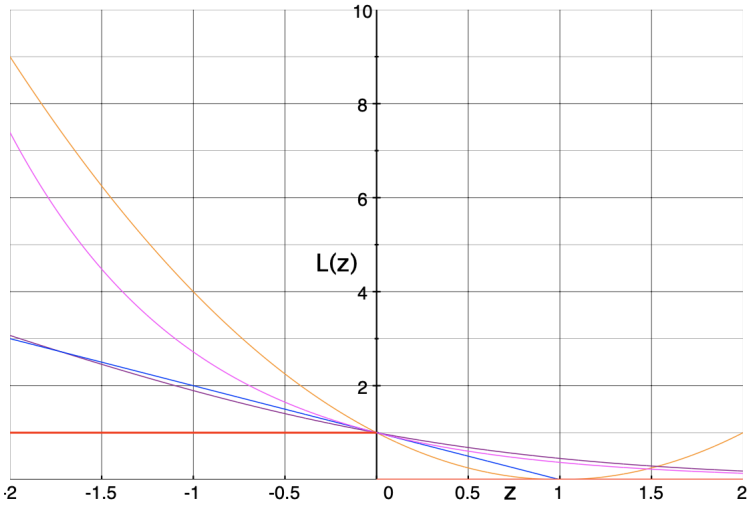


- **squared loss**,

$$L_{sq}(z) = (1 - z)^2 \tag{4.8}$$



Viste tutte in un unico grafico risultano l'una l'approssimazione dell'altra:



4.2 Ranking

Il classificatore rimane binario, ma in output v**oglio classificare le classi dalla più alla meno probabile**. Ovviamente dopo una scoring function, è facile costriure una ranking function.

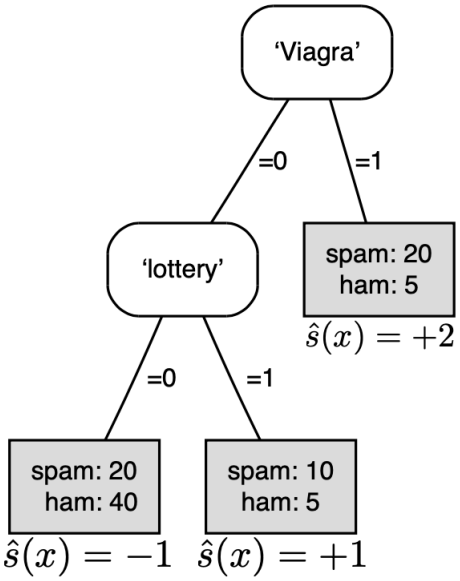
Come valutiamo la bontà del ranking? Possiamo usare il **ranking error rate**, definito come:

$$rank - err = \frac{\sum_{x \in Te^+, x' \in Te^-} I[\hat{s}(x) < \hat{s}(x')] + \frac{1}{2} I[\hat{S}(x) = \hat{s}(x')]}{Pos \cdot Neg} \tag{4.9}$$

x è l'esempio positivo e x' è quello negativo; se lo score x è più piccolo dello score di x' (quindi sto ordinandoli in modo sbagliato, dovrei mettere prima gli esempi positivi e poi i negativi), **aggiungo un punto d'errore** allo score che sto calcolando. **Aggiungo 0.5 punti se gli score dei due esempi sono uguali**; in questo caso non è un errore vero e proprio ma si conta come mezzo punto di errore.

Un **ranking perfetto** corrisponde ad un numeratore uguale a 0 mentre se un **ranking completamente sbagliato** è allora la parte che conta gli uguali è 0 ma l'altra è sempre vera (per ogni coppia possibile, quindi per $Pos \cdot Neg$ coppie), quindi il ranking error sarebbe uguale ad 1.

Facciamo un esempio:



Chiameremo:

- foglia 1, la foglia che ha spam:20, ham 5;
- foglia 2, la foglia che ha spam:20, ham 40;
- foglia 3, la foglia che ha spam:10, ham 5;

Cominciamo dalla foglia 1, abbiamo che quei 5 esempi negativi (ham) sono classificati in una posizione più alta rispetto ai 20 positivi (spam:20) della foglia 2 e anche rispetto ai 10 (spam:10) della 3. Lo diciamo perchè hanno uno score maggiore. Quindi $5 \cdot 10 = 50$ (per la foglia 3), $5 \cdot 20 = 100$ (per la foglia 2) $= 150$ punti di errore.

I 5 positivi della foglia 3 sono messi più in alto dei 20 positivi della foglia 2, quindi 100 punti di errore.

Infine calcoliamo i mezzi punti; sono 475 divisi in $20 \cdot 5 = \frac{100}{2}$ per la foglia 1, $20 \cdot 40 = \frac{800}{2}$ per la foglia 2 e $10 \cdot 5 = \frac{50}{2}$ per la foglia 3.

Abbiamo un totale di $475 + 100 + 150 = 725$ errori su 2500 esempi.

4.3 Stima delle probabilità

Questo è un caso simile a quello dello scoring ma qui mettiamo un vincolo ulteriore agli score, imponendo che siano tutti posiviti e la loro somma sia 1. In simboli

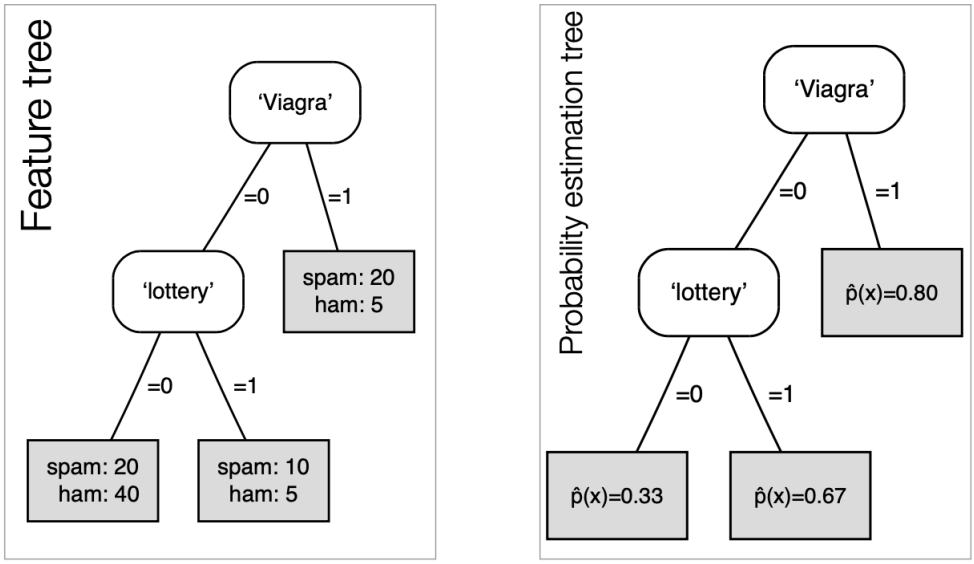
$$\hat{p} : X \to [0,1]^k \tag{4.10}$$

con

$$\hat{p}(x)(\hat{p_1}(x), \ldots, \hat{p_k}(x)). \tag{4.11}$$

Se abbiamo solo due classi, $\hat{p}(x)$ denota la probabilità stimata per la classe positiva.

Probability estimation tree. Per assegnare le probabilità alle foglie, prendiamo gli esempi che ricadono in quella foglia e diciamo che la probabilità che un esempio capiti in quella foglia è data dal rapporto tra gli esempi cercati e la somma degli esempi ($\frac{20}{20+5}$ per la prima foglia).



Come valuto la bontà delle probabilità emesse? Uso la **mean squared probability error**, definita così:

$$\begin{aligned} SE(x) &= \frac{1}{2} ||\hat{p}(x) - I_c(x)||_2^2 \\ &= \frac{1}{2} \sum_{i=1}^{k} (\hat{p}_i(x) - I[c(x) = C_i])^2 \end{aligned} \tag{4.12}$$

dove $I_{c(x)}$ è un vettore che ha 1 nelle posizioni corrispondenti all'etichetta $c(x)$ e 0 nelle altre.

Esempio: assumiamo $\hat{p}(x) = (0.7, 0.1, 0.2)$ e $c(x) = C_1$ e $I_{c(x)}(1,0,0)$. $SE(x)$ sarebbe valutata come:

$$\begin{aligned} SE(x) &= \frac{||(0.7, 0.1, 0.2) - (1,0,0)||_2^2}{2} \\ &= \frac{||(-0.3, 0.1, 0.2)||_2^2}{2} \\ &= \frac{0.09 + 0.01 + 0.04}{2} \\ &= \frac{0.14}{2} = 0.07 \end{aligned} \tag{4.13}$$

Quando lavoriamo su un intero dataset facciamo semplicemente la media degli squared error per ogni istanza

$$MSE(Te) = \frac{1}{|Te|} \sum_{x \in Te} SE(x) \tag{4.14}$$

Stima delle probabilità empiriche. Dato un insieme s di esempi etichettati, possiamo stimare la probabilià delle varie classi creando il vettore $\frac{n_i}{S}$ che è il numero degli esempi etichettati con l'etichetta n_i diviso la cardinalità dell'insieme:

$$\dot{p}(S) = \left(\frac{n_1}{|S|}, \ldots, \frac{n_k}{|S|} \right) \tag{4.15}$$

Correzione di Laplace. Se alcune classi sono poco frequenti e abbiamo pochi esempi di esse, è probabile che la stima calcolata poco fa venga 0. Questo è spesso un problema perchè spesso si moltiplicano tutte le probabilità e avere uno 0 rovina tutto. Quindi cerchiamo di fare "smoothing" di queste probabilità, attraverso la correzione di Laplace. L'idea è che assumiamo di aver pescato un esempio in più per ogni classe:

$$\dot{p}_i(S) = \frac{n_i + 1}{|S| + k} \tag{4.16}$$

Possiamo anche non applicare uno smoothing uniforme, se in partenza sappiamo che non tutte le classi sono equiprobabili:

$$\dot{p}_i(S) = \frac{n_i + m \cdot \pi_i}{|S| + m} \tag{4.17}$$

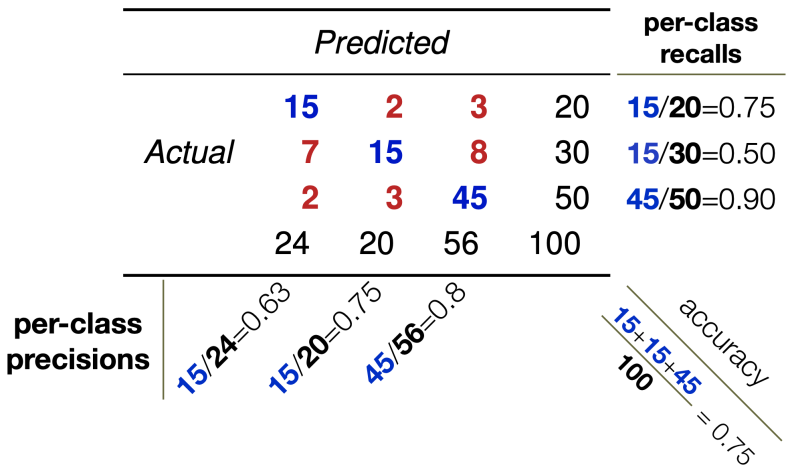
peschiamo m esempi, con $m > k$ e π_i probabilità della classe i-esima.

La correzione di Laplace è un caso speciale della (4.16) con $m = k$ e $\pi_i = \frac{1}{k}$.

Capitolo 5

Classificatori multiclasse

Tavola di contingenza. La tavola di contingenza si ampia per i classificatori multiclasse semplicemente aggiungendo righe e colonne per ogni classe:



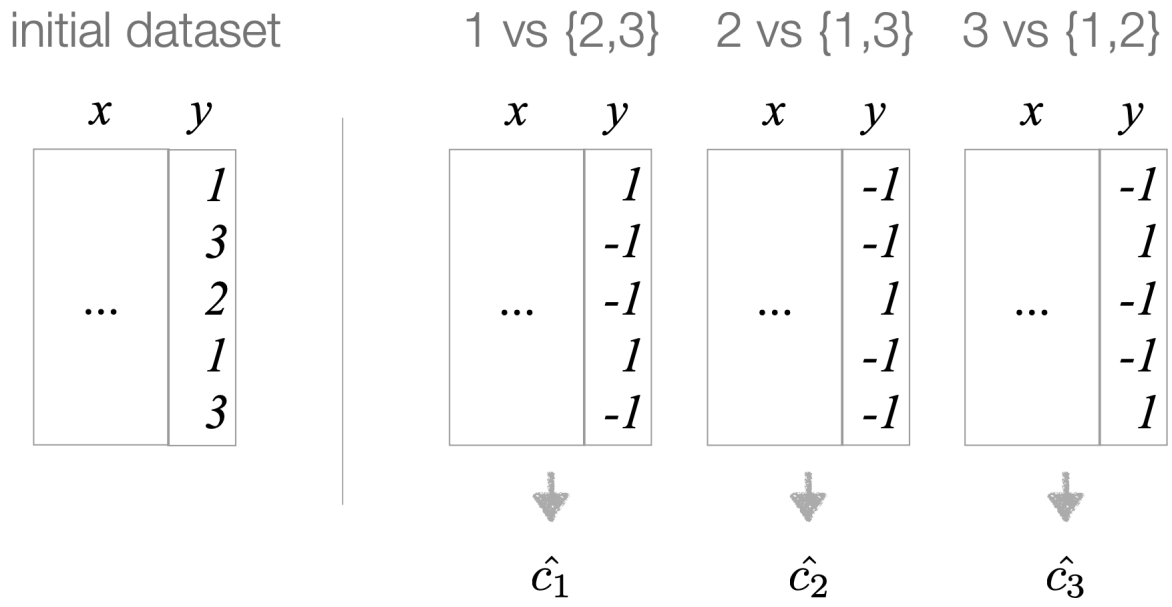
5.1 Estendere classificatori da binari a multiclasse

Ci sono varie tecniche per risolvere un problema multiclasse usando un classificatore binario, tra questi:

- schemi **uno contro tutti**:
  - apprendimento non ordinato,
  - apprendimento ordinato.
- schemi **uno contro uno**:
  - simmetrico,
  - asimmetrico.

L'idea è trasformare il problema di apprendimento, cambiando le etichette del training set e trasformando il problema principale in un certo numeri di problemi binari. Acquisire poi un classificatore binario su questi dataset modificati e poi combinare questi classificatori in un'unica predizione alla fine del processo.

Uno contro tutti (non ordinato) Abbiamo un dataset definito su 3 classi e un certo numero di istanze. Per risolvere il problema, assumiamo di avere a disposizione un algoritmo di apprendimento $L : X^n \to (X \to \{-1, 1\})$, cioè un classificatore binario.



L'idea è quella di costruire la prima volta un dataset che cerca di distinguere la classe 1 dalla 2 e 3, poi un secondo che distingue la 2 dalla 1 e 3 e infine un terzo che distingue la 3 dalla 1 e 2. Facciamo questa operazione prendendo il dataset iniziale e ovunque ci sia scritto 1 restituiamo in output 1, altrimenti -1. Applichiamo poi il nostro algoritmo di apprendimento L al dataset per ottenere il classificatore h_1 e procediamo poi con gli altri due dataset ottenendo h_2 e h_3. A questo punto possiamo passare alla predizione, quindi arriva un nuovo esempio x e bisogna classificarlo correttamente. Questo si fa semplicemente passando il parametro a tutti e 3 i classificatori e ottenendo un vettore con le predizioni (es: $X \xrightarrow{h_1,h_2,h_3} = (1, -1, -1)$). A questo punto è facile da tradurre in una classificazione n-aria.

Questo processo si può formalizzare con una matrice fatta in questo modo:

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \\ vs & vs & vs \\ \{2,3\} & \{1,3\} & \{1,2\} \end{array} \\ \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} \end{array}$$

Questa matrice è utile sia nel momento della codifica che in quello della decodifica: nel momento della codifica le colonne ci dicono come fare il cambio delle etichette perchè, per esempio, la colonna 1 ci dice che, per il classificatore $1vs\{2, 3\}$, l'etichetta c_1 deve essere rinominata +1 e le altre -1. Nel momento della decodifica, andiamo a confrontare il vettore che ci arriva con le righe della matrice e la classe da assegnare al vettore è quella che corrisponde alla riga più vicina al vettore stesso. Da questa matrice si possono definire le altre per gli altri schemi.

Uno contro tutti (ordinato)

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{cc} \textit{1} & \textit{2} \\ \textit{vs} & \textit{vs} \\ \textit{\{2,3\}} & \textit{\{3\}} \end{array} \\ \begin{pmatrix} +1 & 0 \\ -1 & +1 \\ -1 & -1 \end{pmatrix} \end{array}$$

La differenza rispetto allo schema precedente sta nel fatto che qua consideriamo gli output dei classificatori in sequenza, quindi appena trovo un 1 mi fermo. Quindi non valuto tutti i classificatori.

Uno contro uno (simmetrico)

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{ccc} \textit{1} & \textit{1} & \textit{2} \\ \textit{vs} & \textit{vs} & \textit{vs} \\ \textit{2} & \textit{3} & \textit{3} \end{array} \\ \begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix} \end{array}$$

Qui l'idea è che al posto di creare un classificatore che confronta ogni classe con tutte le altre, creo un classificatore per ogni possibile coppia. Si mette 0 quando il modello si astiene.

Uno contro uno (asimmetrico)

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{cccccc} \textit{1} & \textit{2} & \textit{1} & \textit{3} & \textit{2} & \textit{3} \\ \textit{vs} & \textit{vs} & \textit{vs} & \textit{vs} & \textit{vs} & \textit{vs} \\ \textit{2} & \textit{1} & \textit{3} & \textit{1} & \textit{3} & \textit{2} \end{array} \\ \begin{pmatrix} +1 & -1 & +1 & -1 & 0 & 0 \\ -1 & +1 & 0 & 0 & +1 & -1 \\ 0 & 0 & -1 & +1 & -1 & +1 \end{pmatrix} \end{array}$$

L'unica differenza con il precedente sta nel fatto che si considerano anche tutte le coppie inverse (quindi sia $1vs2$ che $2vs1$).

Decoding. Nel caso di uno vs tutti (non ordinato) abbiamo detto che confrontiamo la distanza tra il vettore in input e le righe della matrice, ma come si calcola questa distanza? Così:

$$d(w, c) = \frac{\sum_i (1 - c_i w_i)}{2} \tag{5.1}$$

in cui w è il vettore che contiene gli output dei classificatori binari e d è effettivamente la distanza. Se la classificazione è binaria $c_i w_i$ è 1 se sto classificando correttamente, -1 altrimenti.
Esempio:

	1 vs {2,3}	*2 vs {1,3}*	*3 vs {1,2}*	
				w = (+1 -1 -1)
C_1	$+1$	-1	-1	$d(w, C_1) = 0$
C_2	-1	$+1$	-1	$d(w, C_2) = 2$
C_3	-1	-1	$+1$	$d(w, C_3) = 2$

Cosa succede se per esempio $w = (+1, -1, +1)$? Nel caso di c_1 abbiamo $d = 1$, per c_2 abbiamo $d = 3$ e per c_3 abbiamo di nuovo $d = 1$. Siamo in un caso di ambiguità, come scelgo? Possiamo fare diverse cose: la più banale è prendere a caso uno dei due; un altra soluzione è vedere se questo è un problema diffuso nel dataset e quindi aggiungere colonne così da rendere questo caso ambiguo, più difficile da capitare. Un altro approccio è quello di usare un modello che dia uno score o uno probabilistico.

Difficoltà nell'applicare gli schemi. Non usiamo sempre lo stesso schema perchè non ce n'è uno migliore dell'altro, entrambi hanno i loro vantaggi e i loro svantaggi.

In particolare gli**uno vs tutti** si usano con un dataset non bilanciati, ovvero che non hanno circa lo stesso numero di esempi positivi e negativi.

Gli **uno contro uno** cercano di mitigare il problema assegnando l'etichetta 0 agli esempi che non appartengono alle 2 etichette prese in consederazione. Questo risulta particolarmente problematico quando il dataset è scarso di esempi.