04 - MODEL AND COST FUNCTION (logistic regression)

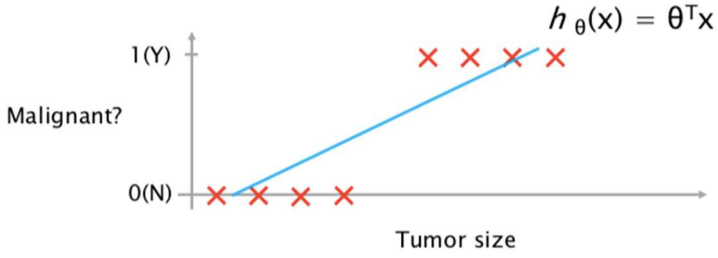
L'algoritmo prende il nome di regressione logistica o abbreviato logit o classificazione.
La classificazione è binaria se si ottiene 0/1 da associare alle classi che si vogliono classificare.

Nota: se sto affrontando un problema di analisi medica esto cercando un tumore maligno

- Classe positiva (1) → trovato tumore maligno
- Classe negativa (0) → non trovato
-

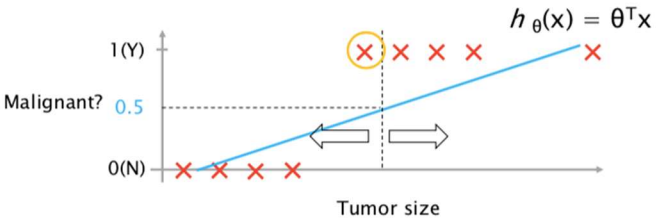
Classification vs Regression

In questi problemi si può tentare un'ipotesi molto azzardata ovvero dare un comportamento lineare.



Potremmo applicare l'algoritmo di regressione lineare già visto e mettere una soglia; quindi, se il valore predetto è maggiore di 0.5 allora y=1 altrimenti y=0.

Sembra perfetto in questo caso ma basta ad esempio aggiungere un valore a destra che non cambia di molto il dataset per dover cambiare l'ipotesi facendo spostare la linea di demarcazione e l'ipotesi non sembra più tanto perfetta.



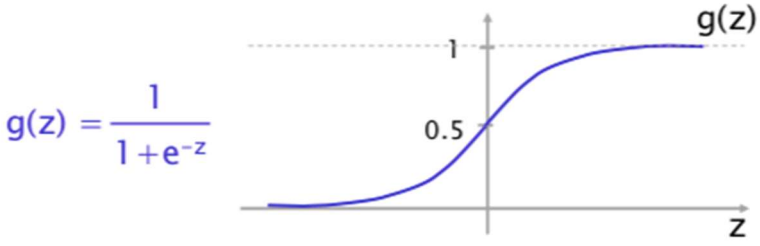
Quindi l'algoritmo di regressione lineare non è pensato per questo tipo di problema.

Bisogna usare un algoritmo di classificazione binaria anche perché con la classificazione lineare si possono avere come output valori > 1 o < 0.

Logistic regression

L'obiettivo è ottenere un'ipotesi che sia in grado di predire un valore $0 <= h_\theta(x) <= 1$

Prendiamo l'ipotesi scritta per la regressione lineare e immaginiamo di applicare a questa ipotesi una funzione g tale per avere una regressione logistica. **$h_\theta(x) = g(\theta^T x)$**
*La funzione g che si utilizza è una **sigmoide** o funzione logistica.*



Ha due asintoti a 0 e a 1 e intercetta le y in 0.5 quindi questa funzione mi mappa una funzione lineare di parametri in un output compreso tra 0 e 1.
In sostanza l'ipotesi lineare non era così sbagliata solo che bisogna mappare i suoi valori nell'output desiderato.

Nelle reti neurali questo è il punto di partenza in quanto sono state sviluppate altre funzioni g, che hanno risultati migliori dal punto di vista dell'addestramento.

Nella regressione logistica l'ipotesi deve essere interpretata come la probabilità che il valore di output assuma valore 1 dato l'input x.

Applicandolo all'esempio di prima se h = 0.7 stiamo dicendo al paziente che ha il 70% di probabilità che il tumore sia maligno.
Formalmente usando la notazione della teoria delle probabilità si ha che

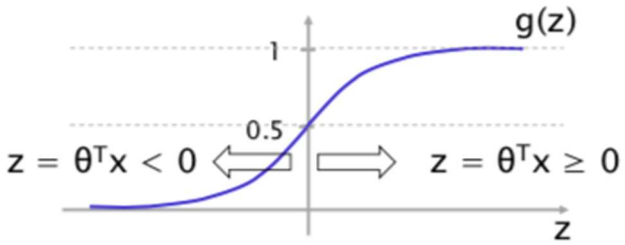
$h_\theta(x) = P(y=1 \mid x; \theta)$ (l'ipotesi è la probabilità che y sia 1 condizionato a x e θ)

e di conseguenza:

$P(y=0 \mid x; \theta) = 1 - P(y=1 \mid x; \theta)$ **--- x; θ vuol dire x parametrizzato in θ**

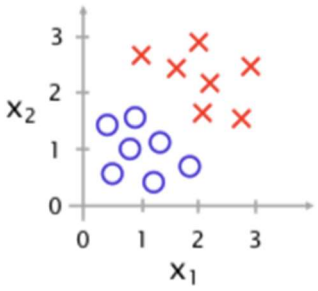
Decision boundary

Quello che ci interessava prima era trovare una sorta di linea di demarcazione per separare gli esempi di una classe da quelli dell'altra.



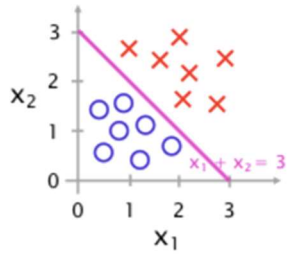
Immaginiamo di avere un problema di classificazione e di avere due classi e avere un problema espresso come due feature x_1 e x_2 e tralasciamo x_0 e il bias.

Immaginiamo che le "X" siano gli esempi positivi e che le "O" siano quelli negativi e c'è un'ipotesi.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Immaginiamo che venga addestrato il nostro algoritmo ottenendo dei parametri (-3;1;1) ottenendo
g(-3 + x1 + x2) quindi prediciamo se il contenuto delle parentesi è >= 0
La regola diventa x1+x2>=3 e se si prende l'equazione x1+x2=3 si ha una retta.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$
$$-3 \quad 1 \quad 1$$
$$h_\theta(x) = g(-3 + x_1 + x_2)$$

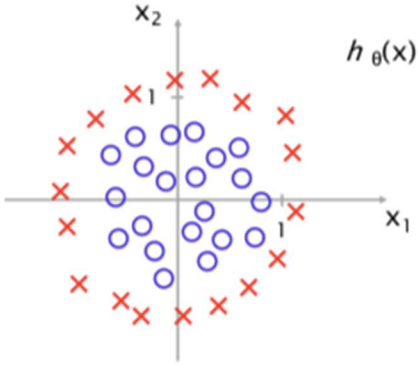
Predict y=1 if $-3 + x_1 + x_2 \geq 0$

$$x_1 + x_2 \geq 3$$

Questa **retta** separa nel semipiano superiore gli esempi postivi e nel semipiano inferiore quelli negativi e viene chiamata il **decision boundary**.

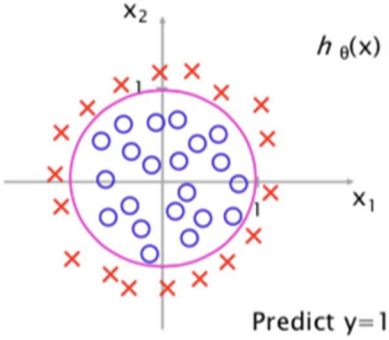
Questa è una proprietà dell'ipotesi inclusi i parametri che la descrivono. Inserendo nuovi valori questi verranno classificati secondo la regola di separazione. Il nostro scopo è trovare i parametri che definiscono il decision boundary. Ovviamente i **parametri possono non essere lineari**.

Si prenda un altro esempio.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

L'ipotesi in questo caso è quadratica. Si avvia l'algoritmo e troviamo i parametri giusti e li mettiamo nell'ipotesi e si ottiene che y=1 se $x_1^2 + x_2^2 >= 1$
Il decision boundary è quindi una circonferenza.



$$-1 \quad 0 \quad 0$$
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$
$$1 \quad 1$$

Predict y=1 if $-1 + x_1^2 + x_2^2 \geq 0$

$$x_1^2 + x_2^2 \geq 1$$

Immaginiziamo che per il problema che abbiamo appena analizzato si possa usare una funzione di costo che è una linear regression.

Cost function (logistic regression)

Il mio vero obiettivo è andare a definire una funzione di costo che serve appunto per trovare i parametri.
Per la regressione lineare avevo una funzione che usava l'errore quadratico medio come distanza tra l'ipotesi e il valore dell'esempio quindi tra il valore predetto e il valore effettivo.

- Linear regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \overbrace{\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2}^{\text{Regression Loss}(h_\theta(x),y)}$$

J(θ) convex

La caratteristica di questa funzione di costo era la sua convessità che aveva un minimo globale.
Nel caso della logistic regression io prendo la mia ipotesi e la inserisco dentro la funzione di costo della regressione lineare.

- Linear regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \overbrace{\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2}^{\text{Regression Loss}(h_\theta(x),y)}$$

J(θ) convex

- Logistic regression

$$h_\theta(x) = g(\theta^T x), \text{ with } g(z) = \frac{1}{1+e^{-z}}$$

J(θ) non-convex

Il risultato è che la funzione di costo sarà **non convessa e quindi ci saranno più minimi locali.**

Possiamo riscrivere la funzione di costo in questo modo, **dividendola in due parti** in modo tale che una osservi la parte positiva e una la parte negativa:

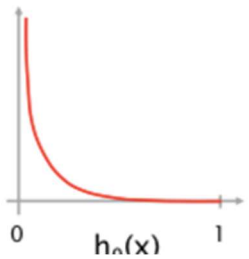
$$\text{Classification Loss}(h_\theta(x),y) \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

(for a single training example)

Si ottiene in questo modo non una regression loss ma una classification loss che è la componente per il singolo training example che mi definisce la funzione di costo.

Se andiamo a plottare questa funzione scopriamo che la funzione ha un andamento del genere cioè che il costo tende a 0 se y=1. Questo ci permette di addestrare la rete in quanto se l'algoritmo predice erroneamente zero penalizziamo l'algoritmo con un costo molto alto.

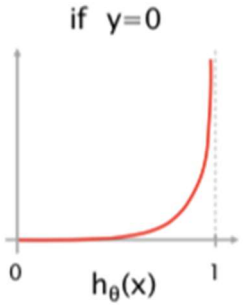
if $y=1$



Loss = 0 if $y = 1$, $h_\theta(x) = 1$

Captures the intuition that if $h_\theta(x) \to 0$ but $y=1$, we'll penalize learning algorithm by a very high cost

Analogamente per y=0 accade un comportamento simmetrico:



Loss = 0 if y = 0, $h_\theta(x) = 0$

Captures the intuition that if $h_\theta(x) \to 1$ but y=0, we'll penalize learning algorithm by a very high cost

Questa riscrittura del loss mi risolverà alcuni problemi.
Se andiamo a prendere l'equazione precedente, gli andiamo a rimettere gli apici, abbiamo la sommatoria sui singoli training example.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Loss}(h_\theta(x^{(i)}),y^{(i)}) \quad \text{(overall)}$$

$$\text{Loss}(h_\theta(x),y) \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$
(for a single training example)

Per risolvere la definizione spezzata in base alla y si può riscrivere l'equazione.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Loss}(h_\theta(x^{(i)}),y^{(i)}) \quad \text{(overall)}$$

$$\text{Loss}(h_\theta(x),y) \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$
(for a single training example)

$$\text{Loss}(h_\theta(x),y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

→Binary cross-entropy cost funzion: **funzione di costo della regressione logistica per tutto il dataset**.
Quindi come per la regressione lineare si ha:

→ **vantaggio:** elimina la componente non lineare (trattando la sigmoide con un logaritmo), riottenendo la convessità della funzione di costo.

- Cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

- To fit parameters θ

  $\min_\theta J(\theta)$, using gradient descent

- To make a prediction given new x

  output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, i.e., $P(y=1 \mid x;\theta)$

- Gradient descent

  repeat until convergence {

  $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

  }

  (simultaneously update all θ_j)

Applicare il gradient descent equivale a calcolare il termine derivativo perché per un passo del gradient descent dobbiamo aggiornare i parametri prendendo quelli del passo precedente – il learning rate per la derivata rispetto al parametro considerato della funzione di costo.

- Gradient descent

  repeat until convergence {

  $\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x^{(i)}$

  }

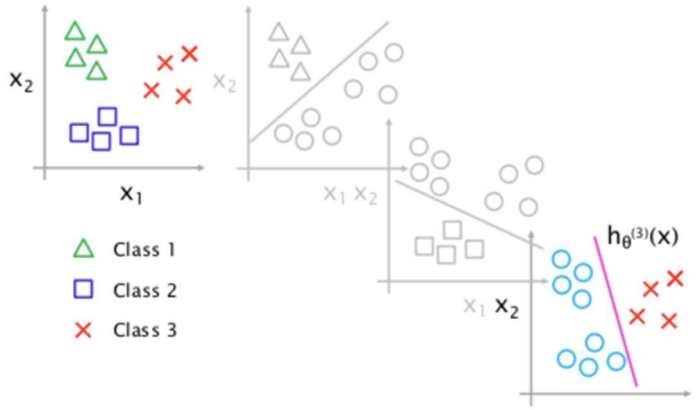
  (simultaenously update all θ_j)

Algorithm looks identical to linear regression!
But it is the definition of $h_\theta(x)$ that has changed

Ha la stessa forma della regressione lineare ma bisogna ricordarsi che h è la sigmoide

Multiclass classification

Esempi di applicazioni : Email tagging: Work, Friends, Family, Hobby; Medical diagnosis: Not ill, Cold, Flu; Weather: Sunny, Cloudy, Rain, Snow.
Un modo per gestire questo tipo di classificazione è il **one-vs-all** dove si prende una classe alla volta e unendo le altre come altra unica classe trovando ***un'ipotesi per ogni sotto-problema***

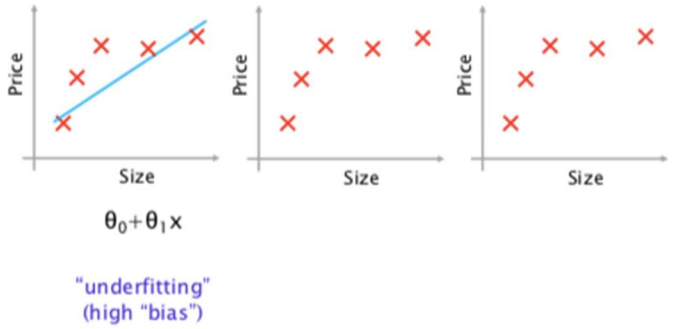


Quindi ci sono n classificazioni binarie. Quando ho un nuovo input prendendo la classe i-esima che massimizza le n ipotesi.

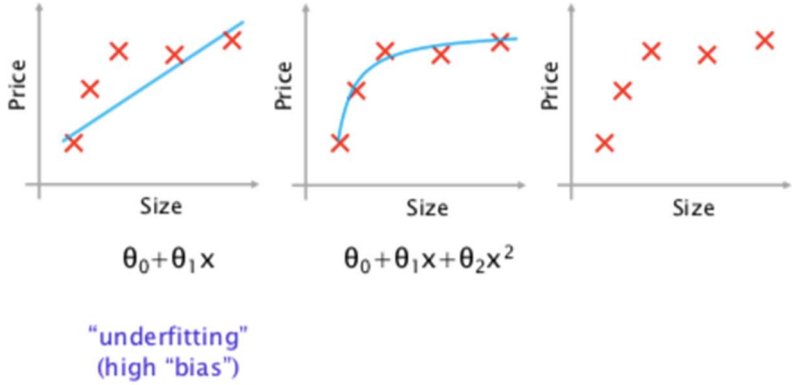
Nota: sono costretto a fare n volte l'operazione.

Il problema dell'overfitting

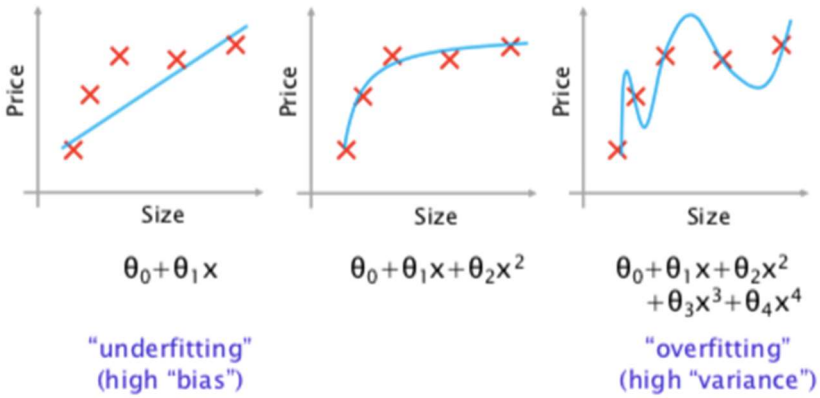
Immaginiamo di aver addestrato un classificatore lineare trovando una retta, l'errore è minimo ma siamo lontani dai dati. Si dice che questa soluzione sta **underfittando** i nostri dati, ovvero che non abbia imparato abbastanza dai dati disponibili. Io vorrei che il mio modello si comportasse bene almeno coi dati di training. Questa situazione si chiama anche **high bias**.



Questo problema si risolve aumentando il grado di complessità del modello aggiungendo gradi al polinomio.
La nuova ipotesi si avvicina al modello di esempio.



Potrei quindi continuare ad aumentare il grado pensando di ottenere risultati sempre migliori e si finisce all'estremo opposto. Riesco ad interpolare tutti i miei dati in una situazione che si chiama **overfitting**

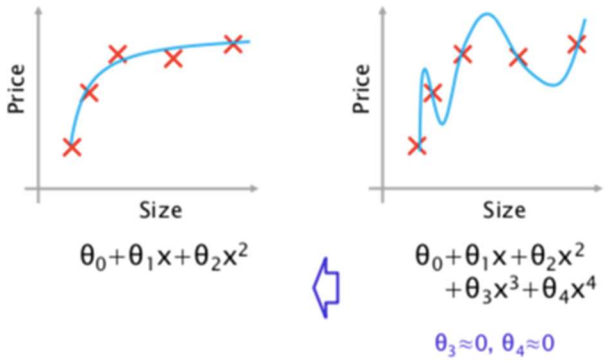


‼ Il problema più grave è l'overfitting perché per l'underfitting mi basta aumentare la complessità del polinomio per migliorare. Mentre con l'overfitting non si riesce più a generalizzare su nuovi esempi che verranno inseriti.

Per affrontare questo tipo di problema posso:

- ridurre il numero di feature
- usare la regolarizzazione tenendo tutte le features (ma ridurre il peso di alcuni parametri.)

Regolarizzazione



Chi è che sceglie come adattare i parametri per la regolarizzazione?
Alla funzione di costo si aggiunge un termine che è una **sommatoria dei parametri al quadrato**. (L2)
Però mentre nella prima sommatoria si somma sui training examples m, sul nuovo parametro si somma sul numero di **features n**. (esempi)
La lambda si chiama **parametro di regolarizzazione** che indica quanto smorzare i parametri.
Il suo valore dipende anche dall'algoritmo di addestramento.

- Small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$
  - "Simpler" hypothesis
  - Less prone to overfitting
- E.g., housing prices
  - Features: $x_1, x_2, \ldots, x_{100}$
  - Parameters: $\theta_0, \theta_1, \ldots, \theta_{100}$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Regularization parameter

→ il nuovo algoritmo si chiama "regressione lineare regolarizzata"
Come faccio in una situazione di overfitting risolvere il mio porblema?

- Se sono un bravo ingegnere che progetta modelli di machine learning: potrei cercare, secondo la mia cultura di dominio, di scegliere le feature da eliminare andando verso un modello più semplice
- Posso aumentare i dati in modo tale che il modello non diventi così complesso
- Regolarizzazione (modifico funzione di costo aggiungendo un altro termine e dovendo scegliere un altro parametro λ)