

Lezione 6

Nicholas Attardo

April 2025

1 Classificazione

1.1 Definizione

La **classificazione** è un processo fondamentale nell'apprendimento supervisionato, il cui obiettivo è partizionare lo spazio delle feature in regioni distinte, ciascuna corrispondente a una classe predefinita. L'assegnazione di un pattern a una classe avviene mediante la creazione di **confini decisionali**, che separano i pattern appartenenti a classi diverse. Questo processo si applica sia alla **classificazione binaria**, in cui si distinguono due sole classi (solitamente indicate come positiva e negativa), sia alla **classificazione multi-classe**, in cui si gestiscono più di due categorie.

Nel caso della **classificazione binaria**, l'obiettivo è dividere lo spazio delle feature in due regioni utilizzando un confine decisionale. Un approccio ingenuo consiste nell'utilizzare un modello di **regressione lineare**, fissando una soglia (ad esempio, 0.5) sull'output continuo del modello per decidere l'appartenenza a una classe. Tuttavia, questo metodo presenta limitazioni significative, tra cui la sensibilità a valori anomali e l'incapacità di vincolare l'output all'intervallo [0, 1]. Per superare queste criticità, si preferisce utilizzare modelli più appropriati, come la **regressione logistica**. In questo caso, l'output del modello è interpretato come la probabilità di appartenenza alla classe positiva, calcolata tramite la funzione sigmoide:

$$h_{\theta}(x) = \sigma(\theta^T x), \quad \text{dove} \quad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Il confine decisionale è definito dai punti in cui tale probabilità è del 50%, ovvero quando $\theta^T x = 0$. Questa equazione descrive un **iperpiano** (una linea in 2D, un piano in 3D, ecc.) che separa le due classi nello spazio delle feature. Ad esempio, con un vettore di parametri $\theta = [-3, 1, 1]^T$, il confine decisionale è rappresentato dall'equazione:

$$x_1 + x_2 = 3,$$

che divide il piano in due regioni corrispondenti alle classi 0 e 1.

Per problemi più complessi, è possibile estendere il modello per ottenere **confini decisionali non lineari** applicando trasformazioni polinomiali alle feature di input. Ad esempio, aggiungendo termini quadratici come x_1^2 e x_2^2 , è possibile apprendere confini circolari o altre forme geometriche complesse, pur mantenendo il modello lineare nei parametri.

Nel contesto della **classificazione multi-classe**, l'obiettivo è dividere lo spazio delle feature in k regioni distinte, ciascuna associata a una delle $k > 2$ classi. Questo può essere realizzato utilizzando strategie basate su **più classificatori binari**, come il metodo **One-vs-All (OvA)** o **One-vs-One (OvO)**. Inoltre, le etichette delle classi sono spesso rappresentate utilizzando il formato **one-hot encoding**, in cui ogni classe è codificata come un vettore con un 1 nella posizione corrispondente alla classe e 0 altrove. La funzione di costo utilizzata per addestrare il modello, come la **cross-entropy loss**, viene generalizzata al caso multi-classe, misurando quanto le probabilità predette dal modello si discostano dalla distribuzione ideale rappresentata dalle etichette vere.

In sintesi, la classificazione è un processo di partizionamento dello spazio delle feature in regioni distinte, ciascuna associata a una classe, mediante la creazione di confini decisionali. Questi confini possono essere lineari o non lineari, a seconda del modello utilizzato e della complessità del problema. La scelta del modello, della funzione di costo e della rappresentazione delle etichette dipende dal numero di classi da gestire e dalla natura dei dati, garantendo una corretta assegnazione dei pattern alle classi.

1.2 Modello Parametrico

Nel contesto dei modelli parametrici per la classificazione, la valutazione delle prestazioni si basa principalmente su due aspetti fondamentali: l'uso della **funzione di costo** e il conteggio degli errori di classificazione. Questi strumenti forniscono una misura oggettiva della capacità del modello di separare correttamente le classi e apprendere dai dati.

La **funzione di costo**, in particolare la **cross-entropy loss**, gioca un ruolo cruciale nella valutazione delle prestazioni durante l'addestramento. Per la classificazione binaria, la cross-entropy loss è definita come:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))],$$

dove $h_\theta(x)$ rappresenta la probabilità predetta dal modello e $y^{(i)}$ è l'etichetta reale del pattern. Questa funzione misura quanto le previsioni del modello si discostano dalle etichette reali, penalizzando fortemente le classificazioni errate. Ad esempio, se l'etichetta reale è 1 ma la probabilità predetta è vicina a 0, il costo aumenta significativamente. L'obiettivo dell'addestramento è minimizzare questa funzione, garantendo una buona corrispondenza tra le previsioni del modello e i dati effettivi. Un valore basso della cross-entropy indica che il modello sta apprendendo efficacemente e che le sue prestazioni sono soddisfacenti.

Oltre alla funzione di costo, un altro modo per valutare le prestazioni è contare **quanti pattern vengono classificati correttamente e quanti no**. Questo approccio fornisce una visione diretta dell'accuratezza del modello, evidenziando gli errori di classificazione e permettendo di identificare eventuali aree di miglioramento.

Le tecniche utilizzate per il **debugging degli algoritmi di regressione**, come l'analisi delle curve dell'errore (J), si applicano anche alla classificazione. L'andamento della funzione di costo durante l'addestramento è un indicatore chiave delle prestazioni e della convergenza del modello. Una diminuzione costante della funzione di costo suggerisce che il modello sta imparando correttamente, mentre plateau o oscillazioni possono segnalare problemi di overfitting o difficoltà di convergenza.

Infine, le fonti menzionano brevemente la possibilità di introdurre una **"rejection option"** per migliorare l'affidabilità del modello. Questa strategia prevede di non fornire una classificazione quando il modello non è sufficientemente sicuro della sua previsione, ad esempio quando la probabilità di appartenenza a una classe è vicina a una soglia di incertezza (es. 0.5). Evitando di classificare in situazioni ambigue, il modello può ridurre il numero di errori e aumentare la sua affidabilità complessiva.

In sintesi, la valutazione delle prestazioni di un modello parametrico per la classificazione si basa sull'analisi della funzione di costo, sul conteggio degli errori e, in alcuni casi, sull'introduzione di una rejection option. La minimizzazione della cross-entropy loss e l'analisi del suo andamento durante l'addestramento sono cruciali per comprendere e migliorare le capacità del modello di separare correttamente le classi. Questi strumenti, insieme a tecniche mutuate dalla regressione, forniscono un quadro completo delle prestazioni del classificatore.

1.3 Classificazione Binaria

La classificazione binaria rappresenta il punto di partenza per comprendere meccanismi più complessi, come la classificazione multi-classe. Mentre nella classificazione binaria l'obiettivo è distinguere tra due sole classi (solitamente indicate come positiva e negativa), nella classificazione multi-classe si estende questo principio per gestire scenari con più di due categorie.

Le fonti evidenziano che un **classificatore multi-classe può essere ottenuto sfruttando più classificatori binari**, utilizzando strategie specifiche che combinano le capacità di modelli binari per affrontare problemi con un numero maggiore di classi. Questo approccio si basa sul principio fondamentale che la comprensione e la costruzione di un modello binario solido costituiscono una base essenziale per risolvere problemi più complessi.

Tra le tecniche principali utilizzate per estendere la classificazione binaria al caso multi-classe, si distinguono:

1. **One-vs-All (OvA):**

- In questa strategia, si addestra un classificatore binario per ogni classe presente nel problema multi-classe.
- Ogni classificatore ha l'obiettivo di distinguere una specifica classe (considerata "positiva") da tutte le altre classi combinate insieme (considerate "negative").
- Ad esempio, in un problema con tre classi ("cerchio", "triangolo", "quadra"):
- Un classificatore distingue "cerchio" da "non cerchio".
- Un secondo classificatore distingue "triangolo" da "non triangolo".
- Un terzo classificatore distingue "quadra" da "non quadra".

2. One-vs-One (OvO):

- In questa strategia, si addestra un classificatore binario per ogni possibile coppia di classi distinte.
- Se ci sono K classi, verranno addestrati $\frac{K(K-1)}{2}$ classificatori binari, ognuno dei quali impara a distinguere tra due specifiche classi.
- Ad esempio, con tre classi ("cerchio", "triangolo", "quadra"):
- Un classificatore distingue "cerchio" da "triangolo".
- Un secondo classificatore distingue "cerchio" da "quadra".
- Un terzo classificatore distingue "triangolo" da "quadra".

Le fonti non entrano nei dettagli su come le decisioni dei singoli classificatori binari vengono aggregate per ottenere la predizione finale nel contesto multi-classe. Tuttavia, il principio chiave è che la classificazione multi-classe si basa sulla solidità e l'affidabilità dei modelli binari. La capacità di costruire e comprendere un buon classificatore binario diventa quindi un passo cruciale per affrontare problemi di classificazione più complessi, garantendo una transizione naturale dalla classificazione binaria a quella multi-classe.

In sintesi, la classificazione binaria non solo rappresenta un problema fondamentale nell'apprendimento supervisionato, ma costituisce anche la base per estendere i concetti a scenari con più di due classi. Le tecniche **One-vs-All** e **One-vs-One** dimostrano come i modelli binari possano essere combinati efficacemente per risolvere problemi multi-classe, sottolineando l'importanza di una solida comprensione della classificazione binaria come prerequisito per affrontare sfide più avanzate.

1.4 Esempi di applicazione

La classificazione binaria rappresenta uno strumento fondamentale nel machine learning, con applicazioni pratiche che spaziano in diversi ambiti del mondo reale. Questo tipo di problema si basa sull'assegnazione di un'istanza a una di due possibili categorie, etichettate come 0 o 1, e si rivela particolarmente efficace per affrontare compiti di discriminazione tra due classi distinte.

Un esempio significativo è il **rilevamento di tumori in mammografie**, in cui l'obiettivo è determinare se una specifica porzione (*patch*) di un'immagine contiene un segno di tumore (classe positiva) oppure no (classe negativa). Questo approccio si basa su dataset accuratamente etichettati da esperti medici e richiede una rappresentazione numerica dei dati che permetta all'algoritmo di distinguere tra le due classi. La complessità del problema può essere affrontata utilizzando tecniche avanzate, come l'analisi multiscala o la combinazione di più classificatori, dimostrando l'importanza della classificazione binaria nella diagnosi medica.

Un altro esempio è il **face detection**, in cui l'algoritmo analizza diverse piccole aree di un'immagine per decidere se contengono un volto (classe positiva) o meno (classe negativa). Anche in questo caso, è necessario risolvere il problema della rappresentazione dei dati, convertendo informazioni visive (come pixel, bordi e texture) in una forma numerica elaborabile dal modello. Il *face detection* è ampiamente utilizzato in settori come la sicurezza e le interfacce utente basate sul riconoscimento facciale, evidenziando l'impatto pratico della classificazione binaria nel riconoscimento visivo.

Il **filtraggio email** (*spam vs non spam*) è un ulteriore esempio di applicazione concreta. In questo contesto, l'algoritmo analizza il contenuto delle email (testo, mittente, oggetto, ecc.) per classificarle come indesiderate (*spam*) o legittime (*non spam*). La qualità della rappresentazione dei dati gioca un ruolo cruciale, poiché caratteristiche come parole chiave frequenti nello spam o la presenza di link devono essere tradotte in una forma numerica che l'algoritmo possa utilizzare per discriminare tra le due classi. Questo esempio sottolinea l'importanza della classificazione binaria nel migliorare l'esperienza utente e proteggere dagli attacchi di phishing.

Infine, la **verifica di notizie** (*fake vs vere*) rappresenta un'applicazione attuale e urgente della classificazione binaria. L'obiettivo è distinguere tra notizie false e reali analizzando il contenuto testuale, la fonte e i metadati. Caratteristiche come l'uso di linguaggio sensazionalistico o la provenienza da fonti poco affidabili vengono estratte e convertite in dati numerici per permettere al modello di prendere una decisione. Questo esempio evidenzia l'importanza della classificazione binaria nel combattere la disinformazione e promuovere una migliore qualità dell'informazione.

In sintesi, la classificazione binaria si rivela uno strumento versatile e potente, applicabile a una vasta gamma di problemi reali. Dalla diagnosi medica al riconoscimento visivo, dal filtraggio email alla verifica di notizie, i principi della classificazione binaria dimostrano la loro rilevanza pratica e il loro impatto sulle sfide quotidiane. Attraverso un'accurata rappresentazione dei dati e l'uso di algoritmi adeguati, è possibile trasformare problemi complessi in soluzioni concrete e utili per il mondo reale.

1.5 Regressione Lineare vs Classificazione

La distinzione tra regressione lineare e classificazione è fondamentale per comprendere le scelte modellistiche nei problemi di machine learning. Mentre la regressione lineare è ideale per predire valori continui, il suo utilizzo per compiti di classificazione binaria presenta limitazioni significative che ne compromettono l'efficacia.

Un approccio ingenuo consiste nell'usare un modello di regressione lineare per stimare un output continuo $h_\theta(x)$ e applicare una soglia (ad esempio, 0.5) per assegnare le classi:

- Se $h_\theta(x) \geq 0.5$, si assegna la classe 1.
- Se $h_\theta(x) < 0.5$, si assegna la classe 0.

Tuttavia, questo metodo si rivela inadeguato per diversi motivi. Innanzitutto, la **sensibilità agli outlier** rappresenta un problema critico. Un singolo punto anomalo può influenzare significativamente la linea di regressione appresa, alterando la decision boundary e causando errori di classificazione su punti precedentemente corretti. Questa instabilità rende il modello poco robusto in presenza di dati non perfettamente puliti.

Inoltre, l'output della regressione lineare non è vincolato nell'intervallo $[0, 1]$. Questo limite è particolarmente problematico in un contesto di classificazione binaria, dove l'output ideale dovrebbe rappresentare una probabilità di appartenenza a una classe. L'output illimitato della regressione lineare rende difficile interpretarlo come una probabilità, compromettendo la validità delle decisioni di classificazione.

Infine, la funzione di costo utilizzata nella regressione lineare (tipicamente l'errore quadratico medio) non è adatta alla classificazione. In questo contesto, l'obiettivo è **massimizzare il numero di pattern classificati correttamente**, richiedendo una funzione di costo specificamente progettata per contare gli errori di classificazione.

Per superare queste limitazioni, è necessario un nuovo modello che soddisfi due requisiti fondamentali:

1. Produrre un output $h_\theta(x)$ compreso tra 0 e 1, interpretabile come una probabilità.
2. Utilizzare una funzione di costo adatta ai problemi di classificazione.

Il modello più appropriato per questi compiti è la **regressione logistica**, che introduce una **funzione sigmoide** ($\sigma(z) = \frac{1}{1+e^{-z}}$) per trasformare l'output di una combinazione lineare delle caratteristiche di input. La sigmoide mappa qualsiasi valore reale nell'intervallo $(0, 1)$, garantendo che l'output del modello sia sempre interpretabile come una probabilità. L'output del modello è dato da:

$$h_\theta(x) = \sigma(\theta^T x)$$

dove $h_\theta(x)$ rappresenta la probabilità che la variabile y sia uguale a 1 dato l'input x e i parametri θ .

In sintesi, mentre la regressione lineare è potente per problemi di predizione continua, il suo utilizzo per la classificazione binaria è limitato dalla sensibilità agli outlier, dall'output non vincolato e dalla mancanza di una funzione di costo adeguata. La regressione logistica, grazie alla sua funzione sigmoide e alla sua capacità di fornire output probabilistici ben definiti, si configura come il modello ideale per affrontare i compiti di classificazione binaria, superando le debolezze intrinseche della regressione lineare.

1.6 Modello di Classificazione (Percettrone)

Nel contesto dei modelli di classificazione, il **percettrone** rappresenta un approccio fondamentale basato su una combinazione lineare delle caratteristiche di input (x_0, x_1, \dots, x_n) pesate dai parametri del modello $(\theta_0, \theta_1, \dots, \theta_n)$. Questa combinazione lineare, espressa come:

$$\theta^T x = \sum_{i=0}^n \theta_i x_i,$$

viene confrontata con una soglia (tipicamente 0) per determinare l'appartenenza di un input a una classe:

- Se $\theta^T x \geq 0$, l'input appartiene alla classe 1.
- Se $\theta^T x < 0$, l'input appartiene alla classe 0.

Tuttavia, il percettrone presenta alcune limitazioni, come la sensibilità agli outlier e la possibilità di produrre output non vincolati all'intervallo $[0, 1]$. Per superare queste criticità, si evolve verso un modello più sofisticato: la **regressione logistica**.

La regressione logistica introduce la **funzione sigmoide** ($\sigma(z) = \frac{1}{1+e^{-z}}$), applicata alla combinazione lineare degli input ($z = \theta^T x$). La funzione sigmoide mappa il risultato della combinazione lineare in un intervallo probabilistico compreso tra 0 e 1:

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

L'output $h_\theta(x)$ può essere interpretato come la **probabilità** che l'input x appartenga alla classe 1. Questa trasformazione permette di ottenere un modello più robusto e interpretabile, riducendo gli effetti negativi degli outlier e fornendo un output continuo che riflette la confidenza nella classificazione.

La **decision boundary** del modello è definita dalla condizione $\theta^T x = 0$, ovvero il punto in cui l'output del modello è 0.5. Questa condizione identifica un iperpiano che separa le regioni di classificazione. Ad esempio, per un modello con due caratteristiche (x_1, x_2) e parametri $\theta = [-3, 1, 1]^T$, la decision boundary è data da:

$$-3 + 1 \cdot x_1 + 1 \cdot x_2 = 0 \quad \Rightarrow \quad x_1 + x_2 = 3$$

In sistemi con risorse computazionali limitate, una volta appresi i parametri θ , è possibile classificare nuovi pattern basandosi direttamente sul segno della

combinazione lineare ($\theta^T x$), evitando il calcolo della sigmoide. Tuttavia, è importante notare che i parametri sono stati appresi considerando l'effetto non lineare della sigmoide durante la fase di training.

Un'altra proprietà cruciale della funzione sigmoide è la sua **derivabilità**, che consente l'utilizzo di algoritmi di ottimizzazione come la discesa del gradiente per apprendere i parametri del modello. In alternativa alla sigmoide, si può utilizzare la **tangente iperbolica** ($\tanh(z)$), che ha un intervallo di output tra -1 e 1, richiedendo una codifica delle etichette di classe come -1 e 1.

In sintesi, il **percettrone** costituisce la base concettuale per modelli di classificazione più avanzati come la regressione logistica. L'introduzione della **funzione sigmoide** migliora significativamente le prestazioni del modello, introducendo non linearità, vincolando l'output a un intervallo probabilistico interpretabile e consentendo l'utilizzo di tecniche di apprendimento basate sul gradiente. Questi elementi rendono la regressione logistica uno strumento potente e flessibile per la classificazione.

1.7 Funzione Sigmoide

La **funzione sigmoide**, definita come:

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

è una funzione fondamentale nei modelli di classificazione binaria, come la regressione logistica. Essa mappa qualsiasi input reale z in un intervallo compreso tra 0 e 1, rendendola particolarmente adatta per interpretare l'output del modello come una **probabilità**. Graficamente, la sigmoide ha una forma a "S", con un asintoto superiore a 1 e un asintoto inferiore a 0. Il valore centrale della funzione è 0.5 quando $z = 0$.

Nel contesto della classificazione binaria, la funzione sigmoide viene applicata all'output di una combinazione lineare delle caratteristiche di input pesate dai parametri del modello ($\theta^T x$). L'output del modello è quindi dato da:

$$h_\theta(x) = \sigma(\theta^T x),$$

dove $h_\theta(x)$ rappresenta la probabilità che la variabile target y sia uguale a 1 dato l'input x e i parametri θ :

$$P(y = 1|x; \theta) = h_\theta(x).$$

Di conseguenza, la probabilità complementare di appartenere alla classe negativa ($y = 0$) è data da:

$$P(y = 0|x; \theta) = 1 - h_\theta(x).$$

Questa relazione deriva dal fatto che, in un problema di classificazione binaria, la somma delle probabilità delle due classi deve essere uguale a 1:

$$P(y = 1|x; \theta) + P(y = 0|x; \theta) = 1.$$

Un aspetto cruciale della funzione sigmoide è il suo ruolo nella definizione del **decision boundary**, ovvero il confine che separa le regioni dello spazio delle caratteristiche associate a ciascuna classe. Il decision boundary si verifica quando l'argomento della sigmoide, $\theta^T x$, è uguale a zero. In questo caso, l'output della sigmoide è esattamente 0.5, indicando un'uguaglianza di probabilità tra le due classi. Matematicamente, il decision boundary è definito dall'equazione:

$$\theta^T x = 0.$$

In uno spazio bidimensionale, questa equazione rappresenta una **linea** che separa le classi, mentre in uno spazio tridimensionale diventa un **piano**, e in generale, in n dimensioni, un **iperpiano**. Ad esempio, con $\theta = [-3, 1, 1]^T$ e $x = [1, x_1, x_2]^T$, il decision boundary è dato da:

$$x_1 + x_2 = 3.$$

Punti con $\theta^T x > 0$ sono classificati come classe 1 ($\sigma(\theta^T x) > 0.5$), mentre punti con $\theta^T x < 0$ sono classificati come classe 0 ($\sigma(\theta^T x) < 0.5$).

Un'altra proprietà fondamentale della funzione sigmoide è la sua **derivabilità**. La derivata di $\sigma(z)$ rispetto a z è data da:

$$\frac{d}{dz} \sigma(z) = \sigma(z) \cdot (1 - \sigma(z)),$$

oppure equivalentemente:

$$\frac{d}{dz} \sigma(z) = \sigma(z) \cdot \sigma(-z).$$

Questa proprietà è essenziale perché consente di costruire una **funzione di loss derivabile**, necessaria per utilizzare algoritmi di ottimizzazione basati sul gradiente, come la **discesa del gradiente**. Questi algoritmi richiedono il calcolo del gradiente della funzione di loss rispetto ai parametri del modello (θ), che è possibile grazie alla derivabilità della sigmoide. Senza questa proprietà, non sarebbe possibile ottimizzare efficacemente i parametri del modello.

Infine, la funzione sigmoide è strettamente legata alla **tangente iperbolica** ($\tanh(z)$), una funzione simile ma con un intervallo di output compreso tra -1 e 1. La relazione tra le due funzioni è espressa come:

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

Questa relazione mostra che la sigmoide può essere vista come una versione traslata e scalata della tangente iperbolica. Sebbene entrambe le funzioni abbiano una forma a "S" e possano essere utilizzate come funzioni di attivazione, differiscono nell'intervallo di output e nelle convenzioni di etichettatura delle classi. Quando si utilizza la tangente iperbolica, le classi devono essere codificate come -1 e 1 invece di 0 e 1. Nonostante queste differenze, entrambe le funzioni forniscono un output limitato e possono essere scelte in base alle esigenze specifiche del problema.

In sintesi, la funzione sigmoide è un elemento chiave nei modelli di classificazione binaria, grazie alla sua capacità di mappare l'output di una combinazione lineare delle caratteristiche in un intervallo probabilistico $(0, 1)$. La sua interpretazione probabilistica, la definizione del decision boundary, la derivabilità e la relazione con la tangente iperbolica ne fanno uno strumento versatile e potente per modellare problemi di classificazione.

1.8 Logistic Regression con Trasformazioni Polinomiali

La regressione logistica standard è un modello lineare che utilizza la funzione sigmoide $h_\theta(x) = \sigma(\theta^T x)$ per stimare la probabilità di appartenenza a una classe. Tuttavia, essendo basato su una combinazione lineare delle caratteristiche, il modello genera decision boundary lineari nello spazio delle caratteristiche, limitandone l'applicabilità a problemi in cui le classi sono separabili linearmente.

Per superare questa limitazione e gestire decision boundary non lineari, si può adottare una strategia simile a quella utilizzata nella regressione lineare: l'applicazione di **trasformazioni polinomiali** alle caratteristiche originali. Questa tecnica consiste nel mappare le caratteristiche originali $x = (x_1, x_2)$ in un nuovo spazio di caratteristiche più ricco, includendo termini polinomiali come x_1^2 , x_2^2 , e $x_1 x_2$. Ad esempio, il vettore delle caratteristiche trasformate diventa $[x_0, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$, dove $x_0 = 1$ rappresenta il termine di bias.

Una volta applicata la trasformazione, il modello di regressione logistica viene addestrato utilizzando il nuovo insieme di caratteristiche trasformate. Formalmente, il modello rimane $h_\theta(x) = \sigma(\theta^T x)$, ma ora x rappresenta il vettore delle caratteristiche trasformate. L'effetto di questa trasformazione è quello di ampliare la capacità del modello, consentendogli di generare **decision boundary non lineari**, come curve o superfici complesse, in grado di separare classi non linearmente separabili.

Un esempio pratico è dato da un decision boundary circolare come $x_1^2 + x_2^2 = 1$. Utilizzando una trasformazione polinomiale, il modello può rappresentare correttamente tale boundary con parametri specifici ($\theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1, \theta_5 = 0, \theta_0 = 1$). In questo caso, il decision boundary circolare emerge naturalmente dallo spazio delle caratteristiche esteso.

L'applicazione di trasformazioni polinomiali aumenta la complessità del modello, permettendogli di apprendere relazioni più elaborate tra le variabili. Tuttavia, questa maggiore flessibilità ha un costo: un numero eccessivo di termini polinomiali può portare al rischio di overfitting, soprattutto quando il dataset è di piccole dimensioni. Pertanto, è fondamentale bilanciare la complessità del modello con la quantità di dati disponibili, eventualmente utilizzando tecniche di regolarizzazione per mitigare il rischio di overfitting.

In sintesi, l'introduzione di trasformazioni polinomiali nella regressione logistica consente di estendere il modello oltre i limiti dei decision boundary lineari, rendendolo adatto a problemi di classificazione più complessi. Questa strategia amplia lo spazio delle caratteristiche e permette al modello di apprendere decision boundary non lineari, migliorando significativamente la sua capacità di

separare classi non linearmente separabili.

1.9 Funzione di Loss (Cross-Entropy)

La **Cross-Entropy Loss** è una funzione di perdita fondamentale nel contesto della classificazione, progettata per misurare quanto le previsioni del modello si discostano dalle vere etichette. Il suo obiettivo principale è penalizzare le previsioni errate o poco confidenti, guidando l'algoritmo di apprendimento a trovare parametri che minimizzino l'incertezza e gli errori di classificazione.

1.9.1 Definizione e Intuizione nella Classificazione Binaria

Nel caso della **classificazione binaria**, la Cross-Entropy Loss è definita separatamente per le due classi ($y \in \{0, 1\}$):

- Se $y = 1$, la loss è $-\log(h_\theta(x))$.
- Se $y = 0$, la loss è $-\log(1 - h_\theta(x))$.

Dove $h_\theta(x) = \sigma(\theta^T x)$ rappresenta l'output del modello, interpretato come la probabilità stimata che $y = 1$. Queste due espressioni possono essere combinate in una formula generale:

$$L(h_\theta(x), y) = -[y \log(h_\theta(x)) + (1 - y) \log(1 - h_\theta(x))]$$

Questa formula cattura l'intuizione che:

- Quando la probabilità predetta ($h_\theta(x)$) si allinea con l'etichetta reale (y), l'errore sarà piccolo (vicino a zero).
- Al contrario, quando il modello fa previsioni errate con alta confidenza, l'errore tenderà a più infinito, penalizzando severamente il modello.

Per valutare le prestazioni su un intero set di training con m esempi, si calcola la funzione di costo $J(\theta)$ come la media degli errori su tutti gli esempi:

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

Questa funzione di costo, chiamata **Cross-Entropy Loss binaria**, è specificamente progettata per problemi di classificazione binaria e viene minimizzata utilizzando algoritmi di ottimizzazione basati sul gradiente, come la discesa del gradiente.

1.9.2 Collegamento con il Concetto di Entropia

La Cross-Entropy Loss è strettamente legata al concetto di **entropia**, una misura di incertezza o disordine in una distribuzione di probabilità. Nella classificazione binaria:

- L'entropia è massima quando la probabilità di appartenere a ciascuna classe è 0,5, indicando la massima incertezza.

Gross Entropy (Loss)

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

Distribuzione data dal classificatore
 $q_1^{(i)} = h_0$
 $q_0^{(i)} = 1 - h_0$

Distribuzione ideale = quelle che vogliono ottenere. Corrisponde alle variabili $y^{(i)}$

$$\begin{aligned} y_1^{(i)} &= 1 & \text{caso } 1 \\ y_0^{(i)} &= 0 & \text{caso } 0 \\ &= 1 - y_1^{(i)} \end{aligned}$$

Misura le "distanze" tra le due distribuzioni

di probabilità p e q

p prob. reale (GROUND TRUTH)

q prob. stima

Distribuzioni in tutti i posti

\Rightarrow vogliono esattamente q così che le due probabilità risultino identiche in tutti i posti
 in input (Minimizzare H)

- L'entropia è minima (zero) quando una delle due classi ha probabilità 1 e l'altra ha probabilità 0.

La Cross-Entropy Loss può essere vista come una misura della "distanza" tra la distribuzione di probabilità reale (data dalle etichette) e la distribuzione di probabilità prodotta dal modello. Minimizzare la Cross-Entropy significa rendere la distribuzione di probabilità predetta il più simile possibile alla distribuzione reale, riducendo così l'incertezza e gli errori di classificazione.

1.9.3 Generalizzazione alla Classificazione Multiclasse

La Cross-Entropy binaria è un caso specifico di una forma più generale utilizzata anche nella **classificazione multiclasse**. La formula generalizzata per la Cross-Entropy Loss multiclasse è:

$$L = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log(\hat{y}_{ik})$$

Dove:

- n : numero di esempi nel set di dati.
- K : numero totale di classi.
- y_{ik} : probabilità reale che l'esempio i appartenga alla classe k , spesso rappresentata in **one-hot encoding**.
- \hat{y}_{ik} : probabilità predetta dal modello che l'esempio i appartenga alla classe k , ottenuta applicando una funzione **Softmax** all'output del modello.

Questa formula confronta la probabilità che il modello assegna a ciascuna classe per ogni esempio con la vera appartenenza di classe, penalizzando fortemente le previsioni errate con alta confidenza. L'obiettivo dell'algoritmo di apprendimento è minimizzare questa funzione di loss, rendendo la distribuzione di probabilità predetta il più simile possibile alla distribuzione reale.

1.9.4 Conclusione

In sintesi, la **Cross-Entropy Loss** è uno strumento potente per la classificazione, sia binaria che multiclasse. Misura quanto le probabilità stimate dal modello si discostano dalle probabilità ideali rappresentate dalle etichette, guidando l'algoritmo di apprendimento a minimizzare l'incertezza e gli errori di classificazione.

Minimizzare la Cross – Entropy Loss significa migliorare la qualità delle previsioni, rendendo la distribuzione

1.10 Apprendimento dei Parametri (θ)

L'apprendimento dei parametri (θ) è un processo fondamentale nell'addestramento di modelli di machine learning. L'obiettivo principale è trovare i **parametri**

ottimali $\hat{\theta}$ che minimizzano la funzione di costo $J(\theta)$, una misura dell'errore compiuto dal modello sull'intero set di training. Nel caso della classificazione binaria, la funzione di costo utilizzata è spesso la **Binary Cross-Entropy Loss**, definita come:

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

dove $h_\theta(x^{(i)}) = \sigma(\theta^T x^{(i)})$ rappresenta la probabilità stimata di appartenenza alla classe positiva.

Per minimizzare $J(\theta)$, si utilizza l'algoritmo di **discesa del gradiente**, un metodo iterativo che aggiorna i parametri nella direzione opposta al gradiente della funzione di costo:

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

dove:

- α : **Tasso di apprendimento (learning rate)**, che scala la derivata e determina la dimensione del passo compiuto verso il minimo.
- Un α troppo basso rallenta la convergenza, mentre uno troppo alto può causare oscillazioni o divergenza.
- $\frac{\partial}{\partial \theta} J(\theta)$: Gradiente della funzione di costo, che indica la direzione di massima crescita della funzione.

La scelta del learning rate è cruciale per garantire una convergenza efficace. Esistono diverse strategie per gestirlo:

1. **Scheduled Learning Rate**: Il tasso di apprendimento viene ridotto gradualmente nel tempo, permettendo all'algoritmo di esplorare rapidamente lo spazio dei parametri all'inizio e di raffinare la ricerca man mano che ci si avvicina al minimo.
2. **Algoritmi Adattivi (es. Adam)**: Utilizzano tassi di apprendimento diversi per ogni parametro, accelerando la convergenza per parametri con gradienti grandi e stabilizzando quelli con gradienti piccoli.

Il monitoraggio della curva dell'errore $J(\theta)$ durante l'addestramento è essenziale per valutare la correttezza del learning rate e identificare eventuali problemi:

- Se la loss diminuisce troppo lentamente, potrebbe essere necessario aumentare il learning rate.
- Se la loss oscilla o aumenta, potrebbe essere necessario diminuire il learning rate.
- Inoltre, l'analisi della curva dell'errore aiuta a rilevare fenomeni come underfitting, overfitting o instabilità.

Una volta trovati i parametri ottimali $\hat{\theta}$, questi possono essere utilizzati per fare previsioni su nuovi dati. La previsione per un input x si basa sul valore:

$$h_{\hat{\theta}}(x) = \sigma(\hat{\theta}^T x)$$

che rappresenta la probabilità stimata di appartenenza alla classe positiva.

In sintesi, l'apprendimento dei parametri (θ) avviene attraverso un processo iterativo di minimizzazione della funzione di costo $J(\theta)$, guidato dall'algoritmo di discesa del gradiente. La scelta del learning rate e l'utilizzo di strategie avanzate per la sua gestione sono aspetti cruciali per garantire una convergenza stabile ed efficiente. Infine, il monitoraggio della curva dell'errore è uno strumento indispensabile per il debugging e la valutazione del modello.

1.11 Decisione e Reiezione

Nel contesto della classificazione binaria, l'apprendimento dei parametri θ gioca un ruolo cruciale per ottenere un modello in grado di stimare probabilità affidabili. Il modello utilizza una funzione sigmoide $h_\theta(x) = \sigma(\theta^T x)$ per produrre un output interpretabile come la probabilità che un input x appartenga alla classe positiva (classe 1). Una volta appresi i parametri ottimali $\hat{\theta}$ attraverso la minimizzazione di una funzione di costo, come la Cross-Entropy Loss binaria, il modello può essere utilizzato per classificare nuovi input confrontando la probabilità stimata con una soglia.

Tradizionalmente, questa soglia è fissata a 0.5: se $h_{\hat{\theta}}(x) \geq 0.5$, l'esempio viene assegnato alla classe 1; altrimenti, viene assegnato alla classe 0. Tuttavia, le applicazioni pratiche spesso richiedono una maggiore flessibilità. In particolare, può essere conveniente introdurre una **soglia di decisione personalizzata** B per gestire situazioni di incertezza. In questo scenario:

- Un esempio viene classificato come classe 1 solo se $h_{\hat{\theta}}(x) \geq B$.
- Viene classificato come classe 0 solo se $h_{\hat{\theta}}(x) < 1 - B$.
- Se la probabilità rientra nell'intervallo $[1 - B, B]$, il modello opta per la **reiezione**, astenendosi dal fornire una classificazione.

La strategia di reiezione è motivata dalla necessità di evitare risposte errate quando il modello non è sufficientemente sicuro delle sue previsioni. Questo approccio è paragonabile al preferire di non rispondere a una domanda piuttosto che dare una risposta sbagliata. Ad esempio, se si utilizza una soglia $B = 0.8$, solo probabilità molto alte ($h_{\hat{\theta}}(x) \geq 0.8$) vengono classificate come classe 1, e solo probabilità molto basse ($h_{\hat{\theta}}(x) < 0.2$) vengono classificate come classe 0. Tutti gli esempi con probabilità intermedie ($0.2 \leq h_{\hat{\theta}}(x) < 0.8$) vengono lasciati non classificati.

La qualità delle stime di probabilità dipende direttamente dai parametri θ appresi durante l'addestramento. Un buon apprendimento dei parametri, ottenuto minimizzando la funzione di loss, porta a stime più accurate e quindi a decisioni di classificazione più affidabili. La scelta della soglia B deve essere guidata dalle specifiche esigenze dell'applicazione, bilanciando il trade-off tra accuratezza e copertura (percentuale di istanze classificate). Una soglia più alta aumenta la sicurezza delle classificazioni ma riduce la copertura, mentre una soglia più bassa include più esempi ma aumenta il rischio di errori.

In sintesi, la fase di decisione nella classificazione binaria può essere arricchita con una strategia di reiezione per gestire l'incertezza. L'implementazione

di questa strategia si basa sull'impostazione di una soglia di probabilità B che riflette il livello di confidenza desiderato. La qualità dei parametri appresi è fondamentale per garantire stime di probabilità affidabili, che influenzano direttamente l'efficacia della strategia di reiezione e, di conseguenza, la qualità delle decisioni prese dal modello.

DERIVATA FUNZIONE COSTO LOGISTIC REGRESSION

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

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

$$= \frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{\frac{\partial}{\partial \theta_j} h_\theta(x^{(i)})}{h_\theta(x^{(i)})} + (1 - y^{(i)}) \frac{\frac{\partial}{\partial \theta_j} (1 - h_\theta(x^{(i)}))}{1 - h_\theta(x^{(i)})} \right] \quad \frac{\partial g(x)}{\partial x} = g'(x) g(x)$$

$$= \frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{\frac{\partial}{\partial \theta_j} \sigma(\theta^\top x^{(i)})}{h_\theta(x^{(i)})} + (1 - y^{(i)}) \frac{\frac{\partial}{\partial \theta_j} (1 - \sigma(\theta^\top x^{(i)}))}{1 - h_\theta(x^{(i)})} \right]$$

$$= \frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{\sigma(\theta^\top x^{(i)}) (1 - \sigma(\theta^\top x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{h_\theta(x^{(i)})} - (1 - y^{(i)}) \frac{\sigma(\theta^\top x^{(i)}) (1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{1 - h_\theta(x^{(i)})} \right] \quad \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$

$$\sigma(\theta^\top x) = h_\theta(x) = \frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{h_\theta(x^{(i)})} - (1 - y^{(i)}) \frac{h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{1 - h_\theta(x^{(i)})} \right]$$

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

$$\frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)}) = x_j^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^m [y^{(i)} - h_\theta(x^{(i)})] x_j^{(i)}$$

cancel
porta dentro segno

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

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

$$\frac{\partial \log(x)}{\partial x} = \frac{1}{x}$$

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

$$\frac{\partial \lg(x)}{\partial x} = \frac{1}{x}$$

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

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

$$h_\theta(x) = \sigma(\theta^\top x) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{\frac{\partial}{\partial \theta_j} \sigma(\theta^\top x^{(i)})}{h_\theta(x^{(i)})} + (1 - y^{(i)}) \frac{\frac{\partial}{\partial \theta_j} (1 - \sigma(\theta^\top x^{(i)}))}{1 - h_\theta(x^{(i)})} \right]$$

$$= -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{\sigma(\theta^\top x^{(i)}) (1 - \sigma(\theta^\top x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{h_\theta(x^{(i)})} - (1 - y^{(i)}) \frac{\sigma(\theta^\top x^{(i)}) (1 - \sigma(\theta^\top x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{1 - h_\theta(x^{(i)})} \right]$$

$$\sigma(\theta^\top x) = h_\theta(x) \Rightarrow -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{(1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{h_\theta(x^{(i)})} - (1 - y^{(i)}) \frac{h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{1 - h_\theta(x^{(i)})} \right]$$

$$= \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)}) = x_j^{(i)} \Rightarrow -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \frac{(1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{h_\theta(x^{(i)})} - (1 - y^{(i)}) \frac{h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} (\theta^\top x^{(i)})}{1 - h_\theta(x^{(i)})} \right]$$

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