INTELLIGENZA ARTIFICIALE

- Riproduzione parziale dell'attività intellettuale propria dell'uomo, per capire se lo fa bene si usa il **Test di Turing**:
    - Criterio oggettivo per valutare se una macchina è intelligente
- **Machine Learning:**
    - Campo di studi che fornisce ai computer la capacità di imparare a risolvere i problemi senza essere esplicitamente programmati
    - Applicazioni:
        - Riconoscimento immagini, vocale, previsione del traffico, guida autono
- **Deep Learning:**
    - Sottoinsieme del machine learning
    - Utilizza le reti neurali per analizzare diversi fattori con una struttura simile al sistema neurale umano.
- Paradigma generale dell'AI:
    - Acquisizione dati
    - Data Processing:
        - Tecniche con cui vengono elaborati i dati
    - Modello:
        - Nucleo principale del sistema AI
        - Insieme di tecniche matematiche e statistiche in grado di apprendere
    - Predizione:
        - Output del modello

Acquisizione dei Dati

- Primo passo cruciale nello sviluppo di un sistema ML
- Si possono ottenere dati in due modi:
    - Usare dataset o database pubblici
    - Acquisendo un nuovo dataset

Annotazione dei Dati

- Una volta acquisiti i dati bisogna **etichettarli**:
    - Contenuto dei dati, può essere numerica o categorica
    - Es: Rilevamento di pedoni: dati: immagini, etichetta: presenza di un pedone
- Se ho dati annotati, il contesto è di learning supervisionato (**supervised learning**)
- Se non ho dati annotati, il contesto è di learning non supervisionato (**unsupervised**)
    - Algoritmo deve imparare a riconoscere pattern nell'input
    - Scopo di apprendere in autonomia la struttura dei dati in ingresso
- Unsupervised learning usato per gestire problemi di clustering

Organizzazione dei Dati

Necessario organizzare i dati come segue:

- **Training set**: dati sui quali il modello apprende
- **Validation set**: vengono messi a punto gli iperparametri
- **Testing set**: dati su cui si testa il modello

Ci sono diversi task in ML a seconda dell'output che vogliamo

Classificazione

- Dato un input, il modello emette una classe
    - Se ci sono 2 classi, il problema è di **classificazione binaria**
    - Se ci sono più classi, **classificazione multiclasse**
- Cos'è una classe (l'output dell'attività di classificazione)?
    - Set di dati con proprietà comuni
    - Es: Classificazione delle lettere italiane: 21 classi
    - Es: email, Classi/etichette: sì/no (spam)

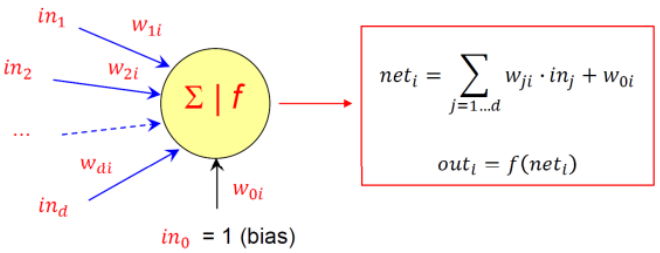
Regressione

- Modella la relazione tra le variabili indipendenti e la variabile dipendente, in modo da poter fare previsioni su nuovi dati.
- Dato un input, il modello restituisce un valore continuo (non una classe)
- Es: Stima dell'altezza di una persona in base al peso

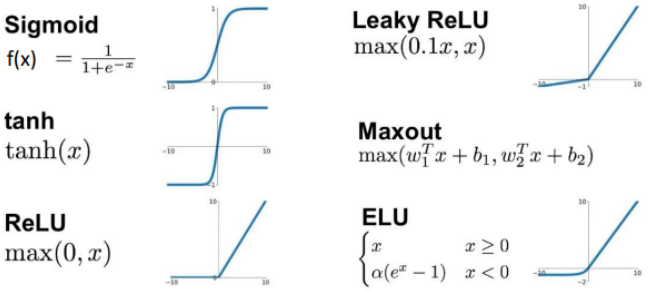
Clustering

- Identificare gruppi (cluster) di dati con caratteristiche simili
- Applicato in un ambiente unsupervised
- Es: Definizione di gruppi di utenti basati sul consumo nel marketing

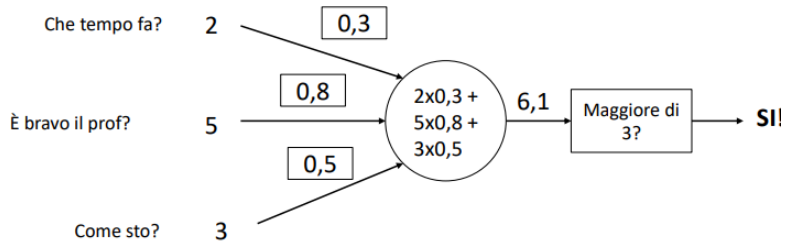
Neurone Artificiale



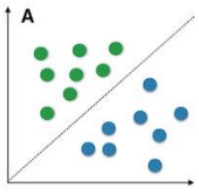
- $in_1, in_2, \ldots, in_d$ sono i **d ingressi** che il neurone i riceve da assoni di neuroni afferenti
- $w_{1i}, w_{2i}, \ldots, w_{di}$ sono i pesi (**weight**) che determinano l'efficacia delle connessioni sinaptiche dei dendriti l'importanza dell'input i-esimo sull'output.
- w_{0i}(detto **bias**) è un ulteriore peso che si considera collegato a un input fittizio con valore sempre 1 questo peso è utile per «tarare» il punto di lavoro ottimale del neurone.
- $f(\cdot)$ è la **funzione di attivazione** e determina se un neurone deve essere attivato o meno. Deve essere non lineare e derivabile

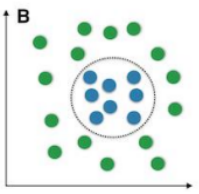


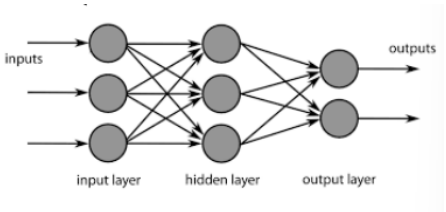
Stamattina esco per andare alla lezione?



- Per poter avere separazioni più complicate:
  - Si utilizzano più neuroni artificiali organizzati su diversi strati (**layer**) **Multilayer Perceptron** (**MLP**)
  - MLP sono fully-connected



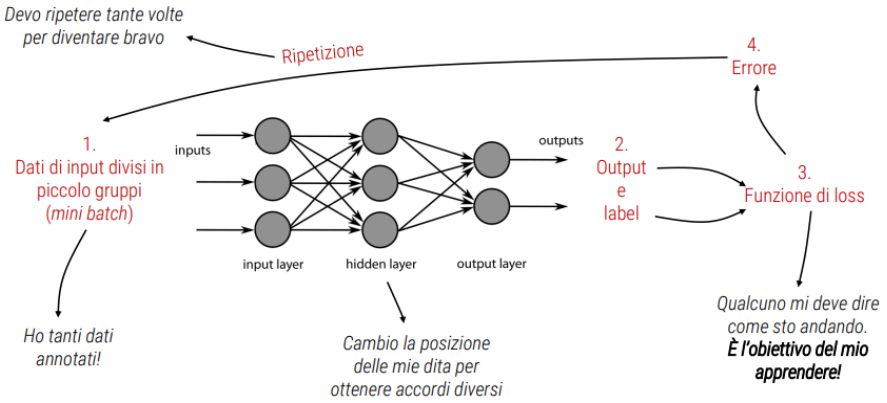




Tipologie di Reti Neurali

- Reti neurali composte da gruppi di neuroni artificiali organizzati in livelli
- Presenti un livello di input, un livello di output, e uno o più livelli intermedi
- Ogni livello contiene uno o più neuroni. I layer intermedi sono chiamati hidden layer in quanto restano “invisibili” dall’esterno della rete
- Feedforward (FFNN):
    - Connessioni collegano i neuroni di un livello con i neuroni di un livello successivo.
    - Non sono consentite connessioni all’indietro
- Ricorrenti:
    - Previste connessioni di feedback e anche all’indietro
    - Crea una sorta di memoria

Addestramento di una Rete Neurale



APPRENDERE = **MINIMIZZARE** LA FUNZIONE OBIETTIVO(derivabile)

Loss Function

- Misura dell'errore della previsione prodotta da un modello di machine learning rispetto ai dati di training

La scelta della loss function dipende dal problema:

- Se problema di classificazione binaria: **Binary Cross-Entropy**

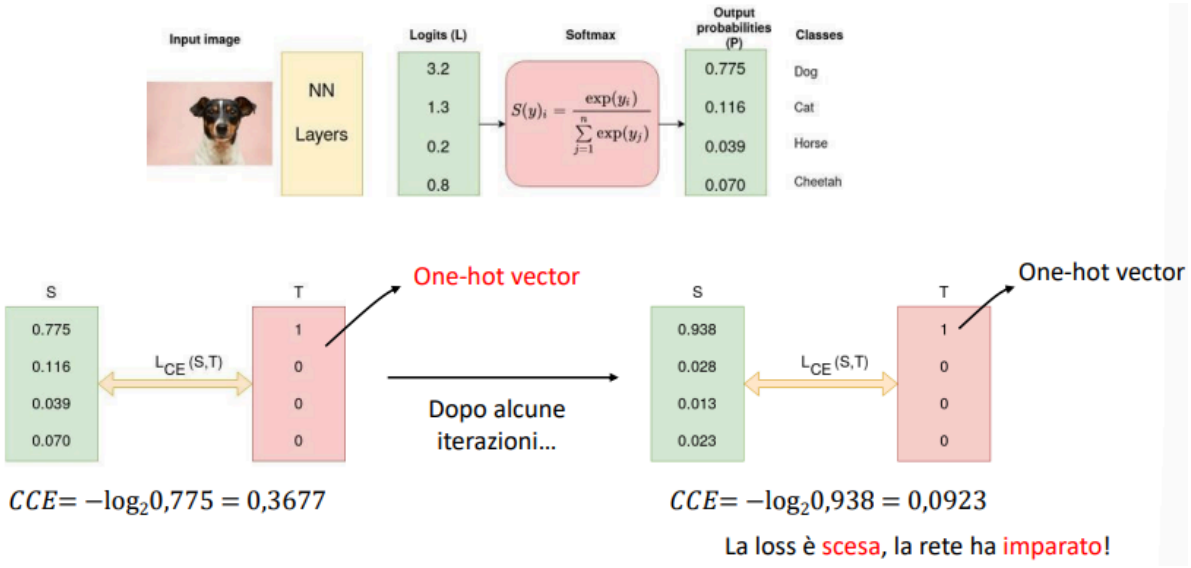
$$BCE = -y_i \cdot \log \hat{y}_i - (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

- Se problema di classificazione multiclasse: **Categorical Cross-Entropy**

$$CCE = -\sum_i y_i \cdot \log \hat{y}_i$$

$\hat{y}_i$ è l’i-esimo valore scalare emesso dal modello (predizione), y_i è la corrispondente label

- Per usare Cross-Entropy Loss, nel layer di output dobbiamo avere delle probabilità
- Si usa il **softmax layer** per trasformare output della rete in probabilità
- Funzione continua e differenziabile



Nel caso di un task di regressione, la loss function più comune è:

- Errore quadratico medio (**Mean Squared Error** o **MSE**)
    - Media dei quadrati delle differenze tra output del modello e output reale

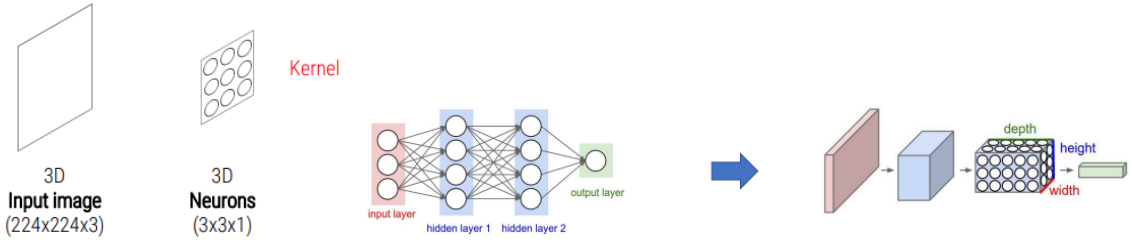
$$C(W) = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i(W))^2}{n}$$

Dove **n** è il numero di esempi di training, $\mathbf{y}_i$ output reale associato a ciascun esempio di training e $\mathbf{\hat{y}_i(W)}$, che dipende dai parametri della rete **W**, è l'output del modello

Convolutional Neural Network (CNN)

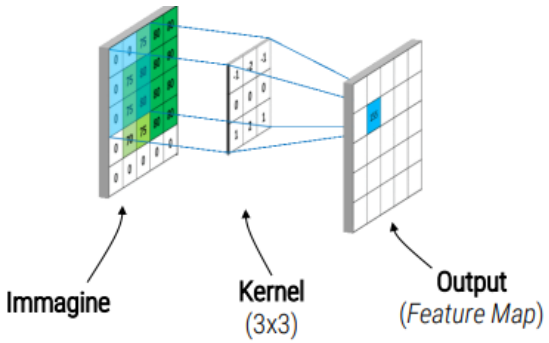
Per l'elaborazione di immagini le reti MLP non sono adatte in quanto sarebbe necessario prevedere un neurone per ogni pixel dell'immagine. Le MLP non hanno alcuna invarianza per traslazione.

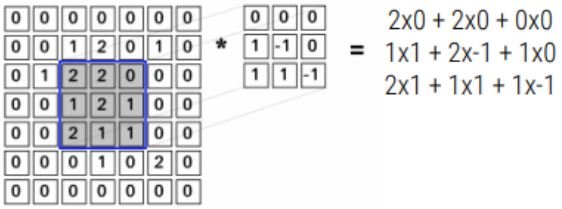
- Le **CNN** sono reti disegnate apposta per processare immagini
- Hanno una struttura a 3 dimensioni



Per collegare il kernel all'immagine si applica la **Convoluzione** (operaz. matematica)

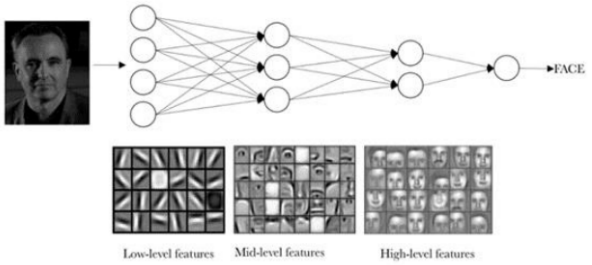
- Una delle più importanti operazioni di image processing attraverso la quale si applicano filtri digitali, per estrarre feature dalle immagini
- Un kernel viene fatto scorrere su ogni pixel di un'immagine di input, per ogni posizione viene generato un valore di output, eseguendo il prodotto scalare tra la maschera e la porzione dell'input coperta.
- Ogni kernel è convoluto con i dati in input, generando **feature map**

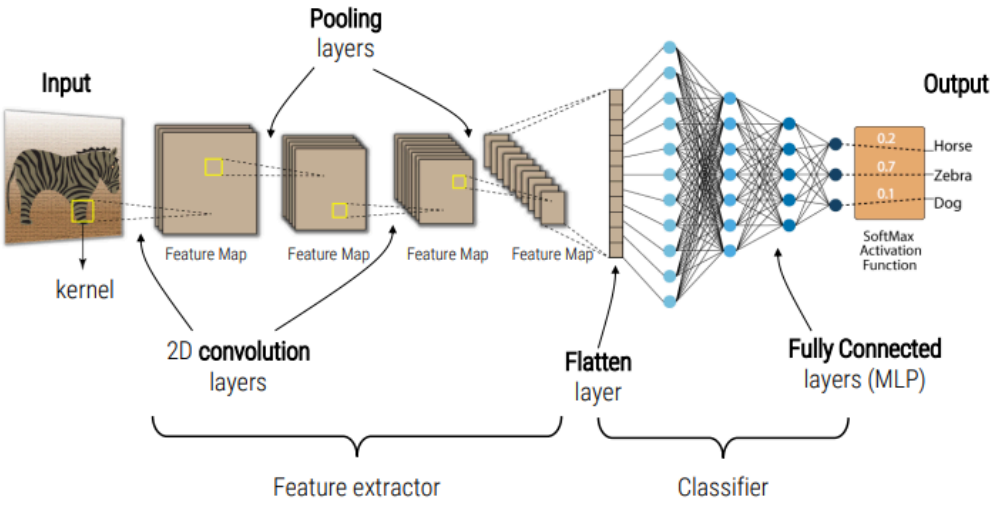




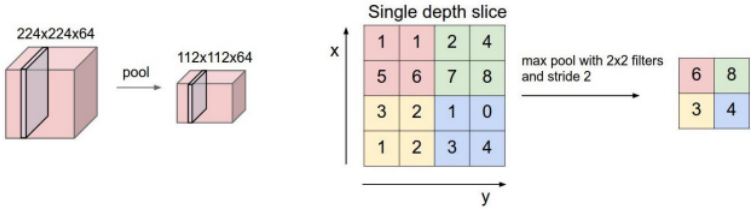
Generalmente, se la profondità è > 1, si hanno più kernel. Una feature map per ogni kernel

- I layer convolutive vicini all'input estraggono info semplici, lontani info complesse

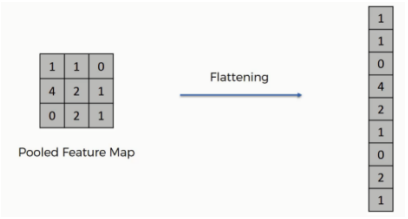




- **Pooling layer** riduce la dimensionalità del volume di input
- Utilizzato per: Invarianza della posizione e Riduzione costo computazionale
- La più utilizzata funzione di pooling è la ***max***



- **Flatten layer** connette il feature extractor con il classifier delle CNN
- “Srotola” il volume di input



- Architetture CNN più note: **VGG**, **ResNet**

Tipologie di training di CNN:

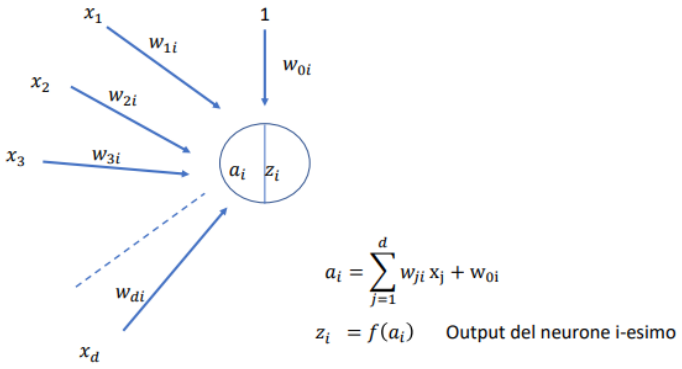
- **From scratch:**
  - Rete addestrata partendo da configurazione random dei pesi
  - Richiesti molti dati di training
  - Training lungo e complesso
- **Pre-trained:**
  - Utilizzata rete già addestrata
  - Output corrisponde a quello originale
  - No training, no bisogno di dati
- **Fine-tuned:**
  - Utilizzata rete già addestrata
  - Vengono congelati i pesi del feature extractor
  - Cambio solo i pesi del classifier

Necessità del training:

- **Big Data**:
  - Disponibilità di dataset etichettati di grandi dimensioni
- **GPU computing**

IA Lazzaro

- Assone e Dendrite, collegamenti tra neuroni, l'assone invia, il dendrite riceve



- a_i è il livello di eccitazione globale del neurone (il potenziale interno calcolato come combinazione lineare degli input pesati (incluso il bias))

Training di una ANN

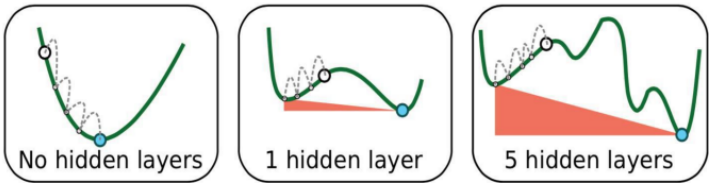
- Dati di addestramento vengono passati attraverso la rete, per calcolare le predizioni
- Confronto tra predizioni della rete e valori desiderati con loss function che misura err.
- Backpropagation: pesi della rete aggiornati nella direzione che minimizza la loss
- Si ricomincia da capo (altra epoca)
- Forward propagation: propagazione delle info in avanti dal livello di input all'output

Problema: definire la complessità del modello. 2 strategie + 1:

- Stabilizzazione Strutturale:
  - Approccio iterativo per trovare l'architettura giusta della rete neurale provandone diverse
  - Rete selezionata è quella con l'errore minimo sul validation set
- Tecnica di Regolarizzazione:
  - Aggiungere alla loss function un termine di penalità sulla norma dei pesi w
  - Ciò restringe l'insieme su cui vengono scelti i parametri
  - Evita che la rete memorizzi il training set e impari l'essenza dei dati
- Early Stopping:
  - Interrompe prematuramente la minimizzazione della loss function
  - Valuta periodicamente l'errore che la rete commette su validation set
  - Addestramento termina quando errore su validation set inizia a aumentare, perché potrebbe iniziare la fase di overfitting della rete a scapito della capacità di generalizzazione

- Minimizzazione dell'errore di training C è un problema difficile poiché:
  - Forti non linearità di C(w) che crea valli ripide o piatte nella loss function
  - Elevata dimensionalità di w ed elevato numero n_T di campioni
  - Presenza di minimi locali non globali

- Aggiungendo hidden layers la loss diventa non convessa e compaiono i min locali



- Metodo di Backpropagation:
  - Algoritmo per calcolare il numero di pesi di un'MLP
  - Per calcolare vettore dei pesi usiamo il metodo del gradiente:

$$\boldsymbol{w}^{(k)} = \boldsymbol{w}^{(k-1)} - \boldsymbol{\eta}\nabla C(\boldsymbol{w}^{(k-1)})$$

  - Dove η è lo step-size (**learning rate**)
  - Bisogna quindi calcolare $\nabla C(\boldsymbol{w})$
  - Ora usiamo la backpropagation per il calcolo del gradiente
  - è uno dei metodi di addestramento più diffusi. Il termine "backpropagation" è legato alla tecnica utilizzata per il calcolo delle derivate della funzione di errore, basata sulle regole di derivazione delle funzioni composte

Tecniche di Ottimizzazione

- BGD:
  - Per calcolo della funzione costo vengono usati tutti i campioni del training set
  - Parametri verranno aggiornati solo una volta per epoca.
  - Funzione costo si riduce uniformemente
  - Costo di calcolo elevato
- SGD:
  - Per calcolo della funzione costo si utilizza una singola osservazione
  - 5 parametri → 5 iterazioni
  - Parametri aggiornati n volte per epoca (n = numero osservazioni)
  - Funzione costo con molte iterazioni
  - Costo di calcolo della funzione costo inferiore a BGD ma tempo più elevato per le iterazioni
- Mini-Batch SGD:
  - Per calcolo della funzione costo si considera sottoinsieme dell'intero set dati
  - Più fluido di SGD
  - Preferibile, buon compromesso tra i due

Iperparametri

- Gli iperparametri sono parametri esterni al modello di machine learning che devono essere impostati prima dell'avvio del processo di addestramento
- Influenzano il comportamento del processo di addestramento
- Elenco:
  - Learning Rate: Quanto velocemente il modello si adatta ai dati
  - Numero di epoche: troppo basso → poco addestrato, alto → overfitting
  - Dimensione Mini-Batch

    - Funzione di attivazione: sigmoid, ReLU

Learning Rate

- Valori bassi:
    - Necessario un numero elevato di passi prima che si completi il training
    - possono far sì che i pesi rimangano bloccati in un minimo locale
- Valori alti:
    - Passa oltre i minimi locali e raggiunge il minimo in meno passi
    - Però si può saltare il minimo target
- Uno degli iperparametri più difficile da regolare

Gradient Descent con Momentum

- Algoritmo di ottimizzazione ampiamente utilizzato per addestrare reti neurali
- Accelera la convergenza e riduce le oscillazioni durante l'allenamento con Gradient Descent.
- Introduce una nuova variabile $v^{(k)}$(velocità) $\boldsymbol{v^{(k)} = \beta v^{(k-1)} + \nabla C(w^{(k)})}$
- Formula aggiornamento pesi: $\boldsymbol{w^{(k+1)} = w^{(k)} - \eta v^{(k)}}$
- Se Beta = 0, classica discesa del gradiente

Learning Rate Scheduling

- Regolazione del learning rate durante l'allenamento è importante tanto quanto la selezione dell'ottimizzatore
- Le tecniche di aggiornamento del learning rate cercano di regolarlo durante l'allenamento riducendolo in accordo ad uno schema predefinito.
- Si utilizzano tecniche di LR decay: riducono gradualmente il LR nel tempo
- 3 tipi di LR decay: (n è il numero di iterazioni eseguite)
    - Step Decay:
        - Riduce il LR di un fattore sigma ogni s epoche
        - $\eta = \eta_0 \cdot \delta^{\left\lfloor \frac{n}{s} \right\rfloor}$
    - Decadimento Esponenziale:
        - $\eta = \eta_0 \cdot e^{-\delta n}$
        - Sigma più alto, più alto decay
    - Decadimento basato sul tempo:
        - $\eta = \frac{\eta_0}{1 + \delta \cdot n}$
        - Modifica LR in funzione del numero di iterazioni eseguite

Metodi di Aggiornamento Adattivo del LR

- modificano dinamicamente il learning rate per ciascun peso durante l'allenamento
- Ottimizzano il ritmo di aggiornamento dei pesi
- Adagrad:
    - Adatta il Learning Rate ai parametri, eseguendo aggiornamenti più grandi per i parametri poco frequenti e aggiornamenti più piccoli per quelli frequenti
    - Elimina la necessità di regolare manualmente il LR
    - Accumulo dei gradienti al quadrato: l'accumulatore di gradiente quadrato può crescere senza limiti, portando a un learning rate troppo piccolo

$$w_j^{(k+1)} = w_j^{(k)} - \frac{\eta}{\sqrt{s_j^{(k)} + \varepsilon}} \nabla C\left(w_j^{(k)}\right)$$

- RMSProp:
    - Deriva da Adagrad e ne migliora alcune limitazioni
    - adatta il learning rate individualmente per ciascun parametro di una rete neurale durante l'allenamento
    - introdotto per ridurre la diminuzione aggressiva del learning rate di Adagrad usando la media ponderata esponenziale dei gradienti al quadrato invece della somma dei gradienti al quadrato
    - Media ponderata esp. può accumulare errori nel tempo

$$w_j^{(k+1)} = w_j^{(k)} - \frac{\eta}{\sqrt{s_j^{(k)} + \varepsilon}} \nabla C\left(w_j^{(k)}\right)$$

- ADAM:
    - Combina le idee di RMSprop e momento per adattare il learning rate in modo individuale per ciascun parametro
    - Adatta in modo dinamico il learning rate per ciascun parametro del modello utilizzando le stime dei momenti del primo e del secondo ordine
    - Converge più velocemente e più robusto
    - $$w_j^{(k+1)} = w_j^{(k)} - \frac{\eta}{\sqrt{\hat{s}_j^{(k)} + \epsilon}} \hat{v}_j^{(k)}$$
- AdaDelta:
    - Adatta il LR in base alla media mobile degli aggiornamenti dei gradienti precedenti
    - A differenza di RMSProp usa una stima della varianza dei gradienti