Relazione OBD

Gianluca Ronzello mat 0354333

Alessio Poggetti mat 0354339

Architettura della rete neurale

La rete neurale implementata è composta da più strati completamente connessi, la cui struttura è flessibile in base ai parametri di input. La configurazione prevede:

- **Input size**: il numero di feature nel set di dati di input.
- **Hidden layers**: la possibilità di specificare uno o più strati nascosti, i cui pesi vengono inizializzati utilizzando l'inizializzazione **He**.
- **Output layer**: configurato per risolvere problemi di regressione o classificazione, con la possibilità di adattare la funzione di attivazione finale (lineare o softmax).

Scelte progettuali e implementative

1. **Inizializzazione dei pesi** Per prevenire il problema del vanishing gradient, i pesi della rete sono stati inizializzati utilizzando la tecnica di **He initialization**. Questa scelta è stata fatta poiché la rete fa uso di funzioni di attivazione ReLU, e l'inizializzazione He assicura che i valori dei gradienti rimangano in una scala accettabile anche con molti strati.

2. **Funzioni di attivazione** La funzione di attivazione scelta per gli strati nascosti è la **ReLU** (Rectified Linear Unit), una delle più utilizzate nelle reti neurali moderne per via della sua semplicità e capacità di risolvere il problema della saturazione dei gradienti. La funzione di attivazione finale varia in base al tipo di problema:

    - **Regressione**: attivazione lineare nell'output, adatta a prevedere valori continui.
    - **Classificazione**: attivazione **Softmax**, che trasforma le uscite in probabilità, utile per i problemi di classificazione multi-classe.

3. **Modularità degli ottimizzatori** Un aspetto chiave del progetto è stato rendere l'implementazione della rete modulare rispetto agli algoritmi di ottimizzazione. In particolare, sono stati creati gli ottimizzatori **Adam** e **Nadam** come classi separate, permettendo di configurare la rete con uno qualsiasi dei due in base alle esigenze. Viene scelto l'algoritmo Adam perché è uno degli ottimizzatori più utilizzati e bilancia i vantaggi della discesa del gradiente con momento e della discesa del gradiente RMSProp. Utilizza una stima adattiva della varianza e del momento del gradiente per aggiornare i pesi, migliorando così la stabilità dell’ottimizzazione. La variante Nadam è stata selezionata in quanto combina i benefici di Adam con l'accelerazione di Nesterov, un approccio che prevede la stima del gradiente "avanzato" prima di compiere l’aggiornamento dei pesi. Questo porta a un’ottimizzazione più veloce, in particolare per modelli con reti profonde.

4. **Regularizzazione L2** Un altro aspetto implementato è stato l’uso della **regularizzazione L2**, utile per prevenire l'overfitting, specialmente in modelli con molti parametri. La penalizzazione dei pesi viene inclusa nel calcolo della perdita, attenuando l'influenza di pesi troppo grandi.

5. **Early Stopping per l'ottimizzazione del training** Durante il training, l'early stopping monitora la performance del modello sul set di validazione, interrompendo l'addestramento se le prestazioni smettono di migliorare dopo un determinato numero di epoche (patience). Questo approccio ha permesso di:
    - Ridurre il rischio di overfitting, fermando l'addestramento non appena il modello inizia a peggiorare su dati non visti.
    - Ottimizzare il tempo di addestramento, evitando che il modello continui a essere allenato inutilmente quando ha già raggiunto il suo potenziale massimo.
6. **Cross-Validation per la ricerca dei parametri** Per garantire l'ottimizzazione dei parametri della rete, è stata adottata la tecnica della **cross-validation**, in particolare la **k-fold cross-validation**. Durante la ricerca dei parametri ottimali, ho utilizzato la cross-validation per determinare:
    - Il learning rate dell’ottimizzatore
    - Il numero di neuroni negli strati nascosti
    - Il parametro di regolarizzazione L2
    - L’ottimizzatore da utilizzare

    Questa tecnica ha permesso di trovare una combinazione ottimale dei parametri che minimizza l’errore su un insieme di validazione, rendendo il modello più robusto e resistente al rischio di overfitting.

Fasi principali dell'implementazione

1. **Forward Pass** La fase di forward prevede il calcolo delle attivazioni e dei valori z attraverso gli strati della rete, usando:
    - **Prodotto matrice-peso**: $Z=X \cdot W+b$
    - **Attivazione ReLU**: per mantenere non linearità tra i livelli della rete.

    Nel caso di regressione, l'output è il valore diretto del layer finale, mentre per la classificazione, l'output è normalizzato tramite la funzione Softmax.
2. **Calcolo della perdita** La funzione di perdita è stata implementata per supportare sia la **regressione** (attraverso l’errore quadratico medio) che la **classificazione** (usando la cross-entropy). Nel calcolo della perdita è inclusa la componente di regularizzazione L2, che penalizza i pesi troppo grandi.
3. **Backward Pass e aggiornamento dei pesi** L'algoritmo di backpropagation calcola i gradienti dei pesi rispetto alla funzione di perdita, e utilizza gli ottimizzatori definiti (Adam o Nadam) per aggiornare i pesi e i bias in base al gradiente calcolato. Questa fase è stata progettata in maniera modulare, consentendo di passare da un ottimizzatore all'altro senza modificare l'architettura della rete.

L'implementazione ha permesso di creare una rete neurale flessibile e adattabile a vari contesti, mantenendo allo stesso tempo la stabilità del processo di addestramento grazie all'utilizzo di tecniche di inizializzazione e ottimizzazione avanzate. La modularità dell’architettura consente una facile estensione della rete per diversi tipi di problema.

Dataset communities and crime

Il dataset combina i dati socioeconomici del censimento statunitense del 1990, i dati delle forze dell’ordine dell’indagine LEMAS statunitense del 1990 e i dati sulla criminalità dell’UCR dell’FBI del 1995.

Gli attributi del dataset sono plausibilmente collegati al crimine e il target è predire i crimini violenti pro capite. Le variabili incluse nel set di dati coinvolgono la comunità, come la percentuale della popolazione considerata urbana e il reddito familiare medio, e coinvolgono le forze dell’ordine, come il numero pro capite di agenti di polizia e la percentuale di agenti assegnati alle unità antidroga. La variabile pro capite dei crimini violenti è stata calcolata utilizzando la popolazione e la somma delle variabili di criminalità considerate crimini violenti negli Stati Uniti: omicidio, stupro, rapina e aggressione.

Il dataset è composto da 127 feature, 1994 istanze, i cui dati sono stati normalizzati. Il problema è di regressione. Nel pre-processamento dei dati sono stati divisi gli attributi tra numerici e categorici. Leggendo le informazioni del dataset si è stabilito che gli attributi categorici non erano utili alla fine della predizione o avevano un numero eccessivo di valori mancanti (>1600), quindi si è optato per rimuoverli.

I parametri migliori trovadi dalla ricerca sono learning rate = 0.001, quattro livelli nascosti da rispettivamente [90, 60, 30, 10] neuroni (più il livello di ingresso e di uscita), parametro di regolarizzazione = 0.001 e come ottimizzatore Nadam

Il risultato viene fornito tramite la radice dell’errore quadratico medio ed è 0.13766

Kuzushiji-49

L’obiettivo del dataset è riconoscere antichi caratteri giapponesi e classificare a quale carattere moderno corrispondono. Kuzushiji-49 contiene 270.912 immagini divise in 49 classi. Nella fase di pre-processamento il dataset è stato normalizzato. Come è stato raccomandato dalle specifiche, come metrica è stata utilizzata l’accuratezza bilanciata.

Dalla cross validation è risultato che i parametri migliori sono: learning rate = 0.001, due livelli nascosti da 500 e 250 neuroni, lambda = 0.01 e come ottimizzatore Nadam

Come risultato si ha un accuratezza del 84,87%