Reti neurali

- **Esponi le reti feed forward**

Una rete feed forward è una rete neurale artificiale con grafo aciclico, il che significa che la computazione procede in modo unidirezionale dai neuroni di input e ai neuroni di output seguendo l'ordine topologico del network.

- In una rete MLP posso collegare un neurone del livello i a uno di livello maggiore di i+1? E nelle FF in generale?

Saltare un livello aiuta a ridurre il problema della scomparsa del gradiente e per indirizzare l'apprendimento in un certo senso ma le reti MLP seguono una rigida suddivisione a strati, quindi, è impossibile saltare uno o più livelli. Questo invece è possibile in un tipo di FF e sono chiamate residual neural network.

- In una rete con due neuroni nello stesso livello, l'uscita del primo può essere l'entrata del secondo e viceversa?

In una feed forward network non è possibile in quanto porterebbe ad avere un ciclo. Per le reti ricorrenti invece è possibile.

- Differenze fra le funzioni di attivazioni sigmoide e gradino

La funzione di attivazione a gradino o step è una funzione lineare e in quanto tale l'output della rete sarà una combinazione lineare (rende quindi complicato risolvere problemi non lineari). Inoltre fare la derivata di una funzione a step non ha molta utilità vista la natura della funzione e quindi non è utile se si vuole usare la discesa del gradiente con error backpropagation.

Una funzione sigmoide, sebbene sia anch'essa una funzione non decrescente, è una funzione non lineare, e si può calcolare facilmente la derivata, tendenzialmente a campana e quindi molto utile per la discesa del gradiente con error backpropagation.

- MLP: definition, training e impieghi

Il multilayer perceptron è un tipo particolare di feedforward network le cui unità di base (i percettroni) sono organizzati in layer e ogni layer ha connessioni solo con il layer successivo. Questo permette di minimizzare il fenomeno delle continue ricomputazioni che avverebbero durante la propagazione del segnale nei normali feed-forward network. La network input function di ogni neurone nascosto e di output viene calcolata come la somma pesata degli input. L'activation function, invece, è una funzione sigmoide, ossia una funzione monotona non decrescente f: R -> [0,1] limx->+inf = 1 lim x->-inf = 0.

La funzione di output è una funzione sigmoide oppure lineare.

La struttura a layer di un MLP suggerisce che si possa descrivere il network con l'aiuto di una matrice dei pesi. In questo modo la computazione del MLP può essere rappresentata attraverso la moltiplicazione tra matrici e vettori.

Il training di un MLP si basa sulla minimizzazione dell'errore. Una delle tecniche molto usate è la tecnica della regressione che ci permette di determinare una relazione funzionale tra il vettore degli input e l'output.
Esistono diversi tipi di regressione: regressione lineare che permette di determinare la retta che devi il meno possibile dai punti del nostro dataset.
Regressione polinomiale e multilineare e regressione logistica.

Un approccio più generale è la tecnica del gradient descent. Il metodo consiste nell'utilizzare la funzione di errore per calcolare la direzione in cui cambiare i pesi e il threshold per minimizzare l'errore. Condizione necessaria per il suo utilizzo è che la funzione sia differenziabile. Tuttavia, solitamente un MLP ha una funzione logistica come funzione di attivazione e, quindi, la funzione di errore sarà differenziabile. Intuitivamente, il gradiente descrive la pendenza di una funzione. Questo è calcolato assegnando ad ogni punto del dominio della funzione un vettore, i cui componenti sono le derivate parziali rispetto agli argomenti. Nel caso delle MLP, calcolare il gradiente della funzione di errore si traduce nel calcolare la derivata parziale della funzione di errore rispetto ai pesi e i threshold presi come parametri. L'errore totale è dato dalla somma degli errori individuali rispetto tutti i neuroni e tutti i training pattern

Se abbiamo come funzione di attivazione la funzione logistica avremo che i cambiamenti operati sul vettore dei pesi saranno proporzionali alla derivata della funzione di attivazione. Più vicini allo o della funzione sono i valori, più ripido sarà il pendio della funzione di errore, per tanto, più rapido l'apprendimento.

Dopo aver trovato l'errore si calcola correzione necessaria per ogni peso e threshold di ogni singolo neurone attraverso il processo di error backpropagation. Si assume che la funzione di attivazione sia la funzione logistica per i neuroni nascosti e di output.
Inizialmente, applichiamo l'input ai neuroni di input che lo restituiscono senza modifiche in output al primo dei layer hidden.
Calcoliamo per ogni neurone dei seguenti layer la somma pesata degli input e al risultato applichiamo la funzione logistica generando così l'output che verrà propagato in tutto il network fino ai neuroni terminali.

A questo punto, calcoliamo la differenza tra l'output atteso e quello attuale e, dato che la funzione di attivazione è invertibile, risaliamo dal vettore di errore a quale fosse l'input che ha condizionato quel particolare errore.
Avendo, ora, trasformato l'errore della variabile di output in quello della variabile di input, possiamo distribuire l'errore e la correzione necessaria in modo proporzionale al ruolo del singolo neurone nel calcolo del seguente output. Propago a ritroso l'errore fino ai neuroni di input. Bisogna osservare comunque che data la forma della funzione logistica l'errore non può sparire completamente, in quanto il gradiente approssimerà il vettore nullo più si avvicinerà allo zero.

Un MLP può essere usato come funzione di approssimazione per funzioni Riemman-integrabili, con 4 strati, o funzioni continue, 3 strati. Vengono chiamati approssimatori universali.

- Self-Organizing Maps: definition, configuration (training) algorithms, operations

Le self-organizing maps sono dei feed-forward network a due layer che possono essere visti come generalizzazione dei LVQN le cui connessioni tra neuroni hidden e neuroni di output sono, però, limitate a quelle tra neuroni "vicini". Come nel caso dei LVQN, la f di input dei neuroni di output è una funzione di distanza tra il vettore di input e quello dei pesi, e la funzione di attivazione è una funzione radiale. Una differenza rispetto ai LVQN è che la funzuione di output è la funzione identità, anche se l'output può essere reso discreto in accordo al principio winner-takes-all, ossia localmente il neurone con la massima attivazione forza a o l'output dei neuroni circostanti. La "vicinanza" tra neuroni è formalizzata assegnando ad ogni coppia di neuroni un numero reale. Queste relazioni possono essere rappresentate graficamente da una griglia bidimensionale. La self-organizing map costituisce una funzione che preserva la topologia, ossia una funzione che preserva la posizione relativa tra i punti del dominio. Il vantaggio nell'usarla è che ci permette di mappare spazi multidimensionali in spazi con dimensioni minori. Come nel caso dei LVQN, il processo di apprendimento si basa sul competitive training: ogni patter in input viene processato ed assegnato al neurone con l'attivazione più alta. Tuttavia, a differenza di quanto accade nell'apprendimento dei LVQN non solo il neurone vincitore viene aggiornato, ma tutti i suoi vicini (sebbene in misura minore). In questo modo si ottiene che i vettori di riferimento di neuroni vicini non si muovano arbitrariamente lontani l'uno dall'altro, mantenendo così la topologia dello spazio di input.

- Reti ricorrenti
Sia gli Hopfield Network che le Boltzmann Machine sono esempi di recurrent network, ovvero network il cui grafo ha al suo interno dei cicli. L'output in questi network viene generato solo se viene raggiunto uno stato stabile nella computazione. L'evoluzione di questi sistemi può essere descritta attraverso

l'utilizzo di equazioni differenziali. Dato, infatti, un insieme di alcune equazioni differenziali rappresentate in forma ricorsiva, possiamo sfruttare la derivata della funzione nell'istante di tempo precedente per calcolare il valore successivo. Questo permette di trasformarle in un recurrent network, creando per ogni variabile un nodo nel grafo e associando alle connessioni il valore del differenziale.

Fuzzy logic
- Membership function, cosa sono e quali sono le differenze tra i vari tipi?

La membership function è strettamente correlata alla teoria Fuzzy poiché attraverso una membership function possiamo definire un Fuzzy Set. Abbiamo diversi modi in cui possiamo interpretare una membership, ma vediamo prima cosa sta definendo. Un insieme fuzzy è un mapping da un insieme di universi X e [0,1], che mappa a ogni elemento dell'Universo un valore compreso tra 0 e 1. La mappatura viene eseguita tramite la funzione di appartenenza, applicando quindi la funzione di appartenenza di uno specifico insieme su un elemento dell'Universo, otteniamo il grado di appartenenza.
La funzione di appartenenza può essere vista in 3 modi diversi:
1) Grado di somiglianza: Stiamo considerando un elemento prototipo e stiamo definendo quanto l'elemento considerato è simile al prototipo.
2) Grado di preferenza: La funzione di appartenenza rappresenta l'insieme degli oggetti più o meno preferiti
3) Grado di possibilità: La funzione di appartenenza U(u) rappresenta la probabilità che la variabile X abbia il valore u
I set Fuzzy e in generale la Fuzzy Theory sono stati creati per affrontare tutte quelle espressioni che gli esseri umani usano tutti i giorni. Non utilizziamo infatti valori netti nei nostri discorsi, utilizziamo termini linguistici imprecisi (es. "Quella macchina è veloce"). La matematica classica e la logica booleana non possono gestire questi valori poiché richiedono precisione e nitidezza. La teoria fuzzy è un modo per affrontare questo tipo di termini. Si parla di variabili linguistiche e valori linguistici. Una variabile linguistica rappresenta un attributo in un sistema fuzzy mentre un termine linguistico è il valore che la variabile può assumere.
Quindi un insieme fuzzy mu di X non vuoto è una funzione dall'insieme di X riferito all'intervallo unitario.

Abbiamo principalmente 2 modi per rappresentare un insieme fuzzy:
1) Rappresentazione verticale: gli insiemi fuzzy sono descritti dalla loro funzione di appartenenza, significa memorizzare il valore della funzione per ogni valore dell'insieme di riferimento
2) Rappresentazione orizzontale: si basa sul concetto degli alpha cuts. Un alpha cut(con alfa compreso tra 0,1) è un insieme contenente tutti quegli elementi che hanno un grado di appartenenza maggiore o uguale ad alfa. Questo insieme è fondamentalmente composto da uno o più intervalli chiusi. La rappresentazione

orizzontale memorizza diversi tagli alfa che rappresentano l'insieme fuzzy. Vale la pena notare che i tagli alfa hanno diverse proprietà come il fatto che il taglio alfa con alfa=0 è l'intero insieme X, se alfa è inferiore a Beta allora il taglio alfa (alfa) includerà il taglio alfa Beta e l'intersezione tra tutti i tagli alfa alfa con alfa inferiore a beta darà come risultato il taglio alfa beta.

La rappresentazione Orizzontale è preferita a quella verticale in quanto è molto più facile da memorizzare in un computer, è composta da intervalli chiusi che possono essere memorizzati tramite liste (contenenti solo gli estremi degli intervalli).
Alcune definizioni utili sugli insiemi fuzzy:
CORE: il core di un Fuzzy Set è composto da tutti quegli elementi che hanno 1 come grado di appartenenza
SUPPORT: è composto da tutti quegli elementi che hanno come titolo di appartenenza un valore diverso da zero
HEIGHT: l'altezza di un insieme fuzzy è il valore massimo del grado di appartenenza a tutti gli elementi dell'insieme di riferimento X. Se l'altezza è uno allora l'insieme fuzzy si dice normale, altrimenti è subnormale.
CONVEX: un insieme fuzzy si dice convesso se tutti i suoi tagli alfa sono convessi
FUZZY NUMBER: è un numero fuzzy se e solo se è normale e la rappresentazione del taglio alfa è limitata, chiusa e convessa per ogni valore di alfa.

Un insieme fuzzy può essere visto anche come una logica multivalore, una logica che non ha solo due valori come quello booleano che è vero o falso, ma una logica in cui abbiamo molti valori diversi di veridicità che è definita dalla funzione di appartenenza. Per avere una logica dobbiamo definire l'equivalente di alcune operazioni ben note nella logica booleana, come 'and', 'or' e 'not'.
Se abbiamo due funzioni di appartenenza u1 e u2 sullo stesso universo X possiamo definire:
(u1 and u2) = min(u1(x), u2(x))
(u1 or uw) = max(u1(x), u2(x))
not u1 = 1 - u1(x)

In questo modo abbiamo un modo per concatenare 2 o più proposizioni fuzzy.
Il complemento può essere definito in molti modi diversi, ci sono alcune condizioni importanti che sono necessarie. La negazione di 0 deve essere 1, la negazione di 1 deve essere 0 e non deve essere crescente. Una negazione è rigorosa se è strettamente decrescente e si dice forte se la negazione della negazione è il valore stesso.

Abbiamo anche bisogno di definire le operazioni di base tra gli insiemi fuzzy che sono l'intersezione e l'unione.
L'intersezione avviene tramite un operatore chiamato T-norma e l'unione tramite un operatore chiamato T-conorm. Queste sono due funzioni prototipiche che hanno

alcune restrizioni ma potrebbero essere qualsiasi funzione si trovi in queste condizioni.
La prima è la legge di identità, quindi la norma T di un valore x con 1 deve essere x e la norma T di x con 0 deve essere x. Le seguenti condizioni sono le stesse sia per la norma T che per la conorma T. La commutatività, l'associatività e la monoticità.
La norma T e la conorma T più utilizzate sono rispettivamente la minima e la massima. Sono le più utilizzate per la loro facile implementazione, ma ci sono altre funzioni che potrebbero essere utilizzate per avere un comportamento leggermente diverso.

Vogliamo anche estendere la definizione di una funzione definita su n insiemi crisp che portano ad un valore di uscita Y ad n insiemi fuzzy che portano ad un insieme fuzzy di uscita mu(y).
L'estensione delle funzioni classiche all'ambiente fuzzy è qualcosa che può essere evitato. Se usiamo come rappresentazione la rappresentazione dei tagli alfa, avremo sostanzialmente degli intervalli. Possiamo usare l'aritmetica degli intervalli per calcolare le operazioni tra tagli alfa e quindi tra insiemi fuzzy, questo è un altro motivo per cui si preferisce la rappresentazione orizzontale.
Anche il concetto classico di relazione crisp tra elementi di due o più insiemi crisp può essere esteso alla relazione fuzzy. Fondamentalmente in una relazione crisp possiamo avere o meno una relazione tra due o più elementi, l'estensione permette di avere più gradi (in [0,1]) di relazione tra gli elementi.
La teoria fuzzy e quindi gli insiemi fuzzy e quindi la funzione di appartenenza viene utilizzata nel controllo fuzzy. È un campo in cui stiamo cercando di usare la teoria fuzzy per controllare un sistema che sta cercando di ottimizzare un problema specifico.
Si differenzia dagli altri tipi di metodi di ottimizzazione perché non sta cercando di modellare il problema e di trovare una soluzione analitica, sta cercando di imitare gli esseri umani che hanno una conoscenza di base, la capacità di vedere la realtà e la capacità di eseguire alcune azioni su esso.
I Fuzzy Controller stanno facendo esattamente questo, stanno osservando la realtà attraverso sensori che forniscono valori crisp, stanno eseguendo la fuzzificazione che sta fondamentalmente convertendo valori crisp in set fuzzy (valori linguistici), stanno applicando una sorta di tabella basata ragionamento e poi, una volta presa una decisione, stanno defuzzificando l'output del sistema per dare un valore netto al sistema controllato che agisce sulla realtà.
È un ragionamento basato su tabelle perché questi sistemi come il Mamdani Controller e il Takagi Sugeno Controller si basano su regole, regole del "se allora", che non sono logica booleana ma logica fuzzy, quindi l'antecedente di una regola (le diverse proposizioni di cui è composta) potrebbe essere vero con diversa veridicità, si prende in considerazione il valore massimo di veridicità tra tutte le regole e si considera la norma corrispondente. Il conseguente viene quindi valutato e un valore di output viene inviato al sistema fisico dopo la defuzzificazione se stiamo usando un

controller Mamdani, se invece stiamo usando un Takagi sugeno non c'è la defuzzificazione, il conseguente della regola è infatti una funzione che ha come output un valore crisp.
La defuzzificazione viene eseguita con metodi diversi che lo sono
1) MAXIMA: prende il massimo del valore di output, poiché il valore di output è un insieme fuzzy, prenderà il valore che ha il grado di appartenenza più alto
2) MOM: media dei massimi, prende l'intervallo composto da tutti quei valori che hanno come grado di appartenenza il massimo, e prende la media di quell'intervallo(i)
3) COG: baricentro: prende il baricentro

FUZZIFICAZIONE/DEFUZZIFICAZIONE

La fuzzificazione è un processo che permette di definire un set fuzzy partendo da un set crisp. Inizialmente il dominio della variabile X è suddiviso in insiemi fuzzy parzialmente sovrapposti, generalmente termini linguistici, per avere una continuità tra di loro, è anche dovuto al fatto che i termini linguistici non sono universalmente definiti, ognuno potrebbe avere una definizione leggermente diversa di un termine linguistico rispetto agli altri.
Una volta definiti i termini linguistici, ogni volta che dobbiamo fuzzificare un input, cioè un valore crisp di una variabile x, dobbiamo calcolare i gradi di appartenenza (dati dalle funzioni di appartenenza dei vari termini linguistici) del valore crisp. La composizione dei vari valori delle funzioni di appartenenza fornirà l'input fuzzificato. Ad esempio, pensiamo che stiamo fuzzificando la temperatura ambientale, se l'input x è 30 gradi, l'input fuzzificato sarà composto dai vari valori delle funzioni di appartenenza in tutti i termini linguistici disponibili, 0 congelamento, 0 freddo, 0 normale, 0,8 caldo, 0,2 molto caldo.
La defuzzificazione viene generalmente calcolata dopo un sistema di inferenza che sta prendendo l'input fuzzificato e prendendo una decisione basata su quello. Questa decisione che è fondamentalmente un output è anche suddivisa in insiemi fuzzy parzialmente sovrapposti (termini linguistici), definiti dalle loro funzioni di appartenenza. La defuzzificazione utilizza concettualmente la grafica dei termini linguisti che rappresentano l'output su un piano bidimensionale (x è il valore, y è il grado di appartenenza). Ogni valore della funzione di appartenenza che si trova nell'intervallo [0,1] viene interpretato come una percentuale e "riempie" la rappresentazione corrispondente nel grafico con quella percentuale specifica. A questo punto ci sono diversi modi per andare avanti, uno di questi è semplicemente prendere il valore di x che sta dando il valore massimo di y (output), il valore defuzzificato è il valore crisp x.
Un altro metodo è il MOM (mean of maxima) che sta prendendo l'intervallo di x valori che portano al valore massimo di y, prendendo la media di quei valori e dando quello come valore di output crisp.

Probabilmente il più utilizzato è il COG (centro di gravità) che calcola il baricentro della regione riempita della rappresentazione grafica dell'output fuzzy. Quest'ultimo metodo è quello preferito perché dà una maggiore continuità nel sistema di controllo (riferito ai passaggi successivi).

Dato un dominio del discorso X, un insieme fuzzy mu è una funzione mu: X->[0,1] che assegna ad ogni elemento un grado di appartenza mu(x) rispetto all'insieme mu.

Queste funzioni sono scelte a seconda del contesto di utilizzo e i gradi di appartenenza sono fissati per convenzione. Possiamo vedere i fuzzy set come interfacce e tra espressioni linguistiche e loro rappresentazioni numeriche.

Ci sono varie semantiche che è possibile associare alla relazione di appartenenza fuzzy a seconda dell'applicazione: somiglianza, preferenza, possibilità.

Nel primo caso, mu(x) può essere vista come il grado di prossimità rispetto ad un elemento prototipale di mu. Questo interpretazione può essere utilizzata nei problemi di pattern classification, cluster analysis e regressione. Nel secondo caso, la funzione mu rappresenta l'insieme degli oggetti preferiti, sia il valore associato ad una decisione X e mu(u) rappresenta l'intensità della preferenza associata ad u, sia la possibilità di scegliere u come valore di X. Questa interpretazione viene utilizzata nei problemi di ottimizzazione fuzzy e nella teoria della decisione. L'ultima delle tre è quella che considera mu(u) come il grado di possibilità che l'elemento u sia il valore del parametro X ed è usata per quantificare lo stato epistemico di un agente. L'obiettivo è quello di distinguere quello che l'agente considererebbe "sorprendente" da quello che, invece, è "tipico" o "aspettato". Questa interpretazione viene utilizzata in data analysis.

- Operazioni in Fuzzy Logic

In un fuzzy set si possono applicare le medesime operazioni relative a un qualsiasi insieme, cioè unione, intersezione e complemento. Per l'intersezione si usa una norma triangolare mentre per l'unione si usa una conorma triangolare, cioè una funzione che soddisfa certe proprietà quali l'identità con 1 e 0 rispettivamente, la commutatività (cioè si possono invertire), l'associatività e il fatto che se $x \leq z$, allora qualsiasi intersezione o unione con x sarà minore o uguale all'intersezione o unione tra x e z. Ci sono diverse possibili funzioni utilizzabili come il minimo per l'intersezione (più forte t-norm) e il massimo per l'unione (più debole t-conorm) come per gli insiemi normali. L'unica proprietà indispensabile per il complemento è

che ¬ 0 = 1 e ¬ 1 = 0. Altre proprietà richieste possono essere che sia strettamente decrescente ($x < y \Rightarrow \neg x > \neg y$), continua e involutiva ($\neg \neg x = x$)

- Quando è preferibile una membership gaussiana a una triangolare?

Una membership function modellata come una gaussiana è preferibile quando si parla di fuzzy number, cioè "quasi 5", e si vuole che un numero abbia un grado di appartenenza alto solo quando è davvero vicino a 5 (data la natura a campana della funzione). Inoltre potrebbe essere necessario considerare la difficoltà di computazione (lineare è più semplice)

- Fuzzy sets: definizione, membership function, operazioni

TUTTO

- fuzzification, defuzzification

Algoritmi evolutivi

- Quale operatore garantisce una maggiore esplorazione dello spazio?
- Evolutionary Algorithms: definizione, operatori

Un problema di ottimizzazione può essere descritto da una tripla (omega, f, <) dove omega è lo spazio di ricerca, f è una funzione di valutazione della forma f: omega -> R e < un preordine. L'insieme H contenuto in omega tale che:
H = {x appartenente a omega | per ogni x' appartenente a omega: f(x) >= f(x')} è definito l'insieme degli ottimi globali. Dato un problema di questo genere la sua soluzione sta nel fornire un elemento che appartiene all'insieme H.
Gli algoritmi evolutivi riescono a risolvere questi problemi adottando una strategia innovativa. Tali algoritmi sono direttamente ispirati alla teoria dell'evoluzione biologica i cui principi fondamentali sono: 1) tratti vantaggiosi che sono risultato di mutazioni casuali tendono ad essere favoriti dalla selezione naturale. 2) gli individui che mostrano questi tratti vantaggiosi hanno migliori opportunità di procreare e moltiplicarsi.

Gli elementi di un algoritmo evolutivo sono:

1) Una codifica per i candidati: dipende molto dal problema e non esistono regole generali.
2) Un metodo per creare una popolazione iniziale: di solito si crea casualmente.
3) Creare una funzione di fitness per valutare i candidati: rappresenta l'ambiente e spesso è la stessa funzione da ottimizzare.
4) Dei metodi di selezione in relazione ai valori di fitness: si scelgono così gli individui che dovranno procreare nella successiva generazione
5) Un insieme di operatori genetici che modifichino i cromosomi: i due più usati sono quello di a) mutazione , che modifica in modo random i cromosomi e

quello di b) crossover che ricombina i cromosomi dei genitori per creare la prole.

6) Alcuni parametri come dimensione della popolazione, probabilità di mutazione
7) Una condizione di terminazione: numero di generazioni, approssimazione all'ottimo, ecc.

Schema generale
Generalmente un'algoritmo evolutivo opera seguendo questi passi:
1. Generazione di una popolazione iniziale
2. Valutazione del fitness degli individui
3. Generazione della nuova popolazione:
    (a) Selezione dei genitori basata sul fitness calcolato al punto precedente
    (b) Generazione di discendenti, partendo dai genitori individuati al punto precedente
    (c) Mutazione randomica degli individui
    (d) Selezione di |pop| individui tra i padri e i figli
4. Ripetizione dal punto 2, fino al raggiungimento del criterio di terminazione

Ragionando sul processo utilizzato, è immediato capire che un algoritmo evolutivo non garantisce la qualità della soluzione trovata ma si spera che, partendo da una popolazione iniziale che copre molte aree dello spazio di ricerca, migliorando iterativamente gli individui si riesca a trovare una soluzione buona.

Gli operatori genetici sono applicati ad una frazione di individui scelti (popolazione intermedia). Vengono così generate mutazioni e ricombinazioni delle soluzioni già esistenti. Gli operatori genetici vengono classificati secondo la loro varietà in: One, two o multiple parent operators.

Nella prima classe possiamo trovare l'operatore di mutazione, il quale introduce piccoli cambiamenti randomici nel genoma della soluzione a cui viene applicato. Risulta utile per introdurre biodiversità nel pool delle soluzioni e favorire l'esplorazione dello spazio di ricerca. Esistono vari metodi per operare una mutazione:
Standard mutation: il valore di uno o più geni viene mutato.
Pair swap: si scambia la posizione di due geni.
Shift: si shifta a destra o sinistra un gruppo di geni.
Arbitrary permutation: si permuta arbitrariamente un gruppo di geni.
Inversion: si inverte l'ordine di apparizione di un gruppo di geni.

Invece, l'operatore di gran lunga più importante tra quelli two-parent è quello di ricombinazione o crossover, il quale ha il compito, date due soluzioni, di creare attraverso una combinazione del loro codice genetico le soluzioni che costituiranno la generazione futura. Vi sono vari modi per operare questa ricombinazione:
One-point crossover: si determina una posizione casuale nel cromosoma e si scambiano le due sequenze da un lato del taglio.
Two-point crossover: si determinano due posizioni casuali nel cromosoma e si scambia quell'intervallo di geni.
N-point crossover: una generalizzazione dei precedenti. Si scambiano le aree incluse nei punti selezionati casualmente.
Uniform crossover: per ogni gene si determina se scambiarlo o meno a seconda di un certo parametro di probabilità.
Shuffle crossover: si procede inizialmente ad operare una permutazione randomica sui due cromosomi. Dopo si procede come nel one-point crossover e si conclude facendo l'unmixing.
Uniform order-based crossover: simile allo uniform crossover, per ogni gene si decide se tenerlo o cambiarlo. Gli spazi sono riempiti nell'ordine di apparizione dei geni dell'altro cromosoma.
Edge-recombination crossover: il cromosoma è rappresentato come un grafo. Ogni gene è un vertice che ha archi verso i suoi vicini. Gli archi dei due grafi vengono mischiati. Si preserva l'informazione relativa alla vicinanza.

Un caso di multiple-parent operator è quello del diagonal crossover. Simile al n-point crossover, ma vi partecipano più di due genitori. Dati k genitori, si scelgono k-1 punti per il crossover e si procede shiftando diagonalmente le sequenze rispetto ai punti scelti. Aumentando il numero di genitori si ottiene un ottimo grado di esplorazione dello spazio. Alcune proprietà che possono caratterizzare gli operatori di crossover sono:

Positional bias: quando la probabilità che due geni vengano ereditati assieme dallo stesso genitore dipende dalla posizione (relativa) dei due geni nel cromosoma. Deve essere evitato perché può rendere la disposizione dei geni cruciale per la riuscita dell'algoritmo.
Distributional bias: quando la probabilità che un certo numero di geni siano scambiati tra i genitori non è la stessa per tutti i possibili numeri di geni. Deve essere evitato perché soluzioni parziali di differenti lunghezze hanno differenti probabilità di progredire alla generazione successiva. In generale, è meno problematico del positional bias.

Per migliorare le perfomance delle mie soluzioni ho due strategie:
Interpolating recombination: opero una fusione dei tratti dei due genitori in modo da creare nuovi discendenti. Si creano nuovi alleli e ne beneficiano particolarmente gli individui con migliore fitness. Per una esplorazione sufficientemente ampia di

omega nelle prime iterazioni occorre utilizzare una probabilità di mutazione molto alta.

Extrapolating recombination: inferisco informazioni da una moltitudine di individui e creo nuovi alleli in accordo. L'influenza della diversità è difficilmente quantificabile.

Domanda 1:
breve introduzione reti neurali con le varie variabili associate ai neuroni e le funzioni, struttura MLP, le funzioni che usa ai vari layer, tutta la fase di training spiegando come funziona la backpropagation e gradient descent con anche le formule associate al singolo aggiornamento dei pesi, due parole sul learning rate, due sulla scomparsa del gradiente e di come è possibile risolverlo, numero di neuroni degli strati hidden (la formula), underfitting e overfitting e il fatto che una mlp con 4 layer può essere usata come approssimatore universale

domanda 2:
breve introduzione del perché vengono usati e a cosa servono, definizione insiemi fuzzy e funzioni membership, forme delle funzioni membership, significati delle funzioni che dipendono dal contesto (similarità, preferenza, possibilità), le definizioni (alfa-cut, support,core, altezza, quando una funzione è normale e quando subnormal), i vari modi di esprimere la disgiunzione la negazione e la congiunzione. Introduzione ai fuzzy controller, com'è composto un controller, Mamdani e takagi

Domanda 3:
introduzione a cosa sono gli algoritmi evolutivi, da cosa è composto un EA, pressione selettiva e time-dependent selective pressure, metodi di selezione (spiegare proprio nel dettaglio), problemi della roulette wheel selection (problema della dominance, affollamento, e problema della varianza con il modo per risolverlo), elitismo, operatori genetici, due parole su ognuno per far capire cosa fanno, i due bias che hanno gli operatori di ricombinazione (distributivo e posizionale), tutto il discorso su "interpolating and extrapolating recombination"

RETI NEURALI RICORRENTI

Le reti neurali ricorrenti sono reti neurali così composte da neuroni che hanno una funzione di input che definisce lo stato di input, una funzione di attivazione che definisce lo stato di attivazione e una funzione di output che definisce l'output del neurone. Essendo una rete i neuroni sono collegati tra loro e ogni connessione è direzionale e ha un peso specifico. Il termine ricorrente definisce la possibilità nella rete di avere connessioni tra neuroni di uno strato e neuroni di strati precedenti, quindi è possibile che un neurone non abbia tutto l'input disponibile nel momento in cui deve calcolare l'output. La differenza tra il feed forward NN e il Recurrent NN è esattamente questa. Nella rete neurale Feed Forward il calcolo segue l'ordine topologico dei neuroni, potremmo pensare di posizionare i neuroni in

un ordine da sinistra a destra e vedere il calcolo andare da sinistra a destra senza mai tornare indietro. In ogni rete neurale avremo i neuroni che iniziano il calcolo ogni volta che ha tutti gli input disponibili, in Recurrent NN potrebbero esserci situazioni in cui un neurone ha bisogno di calcolare l'output anche se non ha tutti gli input disponibili, perché il neurone che fornisce l'input non disponibile dipende dall'output di quel neurone, quindi in una prima esecuzione il neurone deve iniziare il calcolo utilizzando un valore casuale.
Quando abbiamo a che fare con Recurrent NN dobbiamo prendere in considerazione anche l'ordine. In feed forward l'ordine è dato dalla struttura stessa e l'output finale è calcolato quando il calcolo ha raggiunto gli ultimi strati della rete, in Recurrent dobbiamo definire un ordine in cui vengono eseguiti i calcoli e ne parleremo sull'output finale quando raggiungiamo uno stato stazionario, uno stato in cui se viene eseguito un altro calcolo non ci sono cambiamenti nei neuroni. Potrebbero esserci anche situazioni in cui definiamo una condizione di arresto diversa come un numero limitato di calcoli. Ci sono anche casi in cui non raggiungeremo mai uno stato stazionario, la rete continua a cambiare e gli aggiornamenti non convergono mai a uno stato stazionario.
Un altro problema che abbiamo con le reti neurali ricorrenti è il modo in cui dobbiamo calcolare la formazione. In Feed Forward NN siamo abituati ad utilizzare la tecnica della minimizzazione dell'errore quadrato poiché abbiamo un calcolo che è definito e alla fine di quel calcolo possiamo vedere qual è l'errore che si è verificato, anche per i livelli nascosti sappiamo di possiamo usare la back-propagation per stimare l'errore che viene commesso nei livelli nascosti in cui non abbiamo davvero un output previsto. Nelle reti ricorrenti viene utilizzata un'altra tecnica, dispieghiamo la rete nel tempo. Il calcolo è completato, cioè quando raggiunge uno stato stazionario, in quel momento sappiamo quante iterazioni sono state fatte, l'idea è quella di prendere la rete e ripeterla tante volte quante sono le iterazioni che la rete aveva bisogno per raggiungere uno stato stazionario. In questo modo una connessione che stava "tornando indietro" ora è una connessione che va al gruppo successivo della rete, alla fine avremo una rete neurale feed forward a cui possiamo applicare il classico metodo di backpropagation che ' riutilizzato nelle reti feed forward, dovremo finalmente combinare gli aggiornamenti che abbiamo ottenuto.
Le NN ricorrenti sono molto utili per rappresentare e risolvere equazioni differenziali e anche gruppi di equazioni differenziali, utilizzando le cosiddette reti neurali vettoriali. Sono utilizzati in questo campo per la loro particolarità di avere connessioni ricorrenti tra neuroni, utili a rappresentare equazioni differenziali.

Un esempio di Recurrent NN che abbiamo visto sono le reti Hopfield. Si tratta di reti neurali ricorrenti senza neuroni nello strato nascosto e lo strato di input e lo strato di output sono composti dagli stessi neuroni. Ogni neurone è connesso a tutti gli altri neuroni ma non è connesso a se stesso, quindi non ci sono neuroni autoconnessi e i pesi tra di loro dovrebbero essere simmetrici, quindi W(u,v)=W(v,u). La funzione di input di ogni neurone è semplicemente la somma pesata dell'output degli altri neuroni, la funzione di attivazione è una semplice funzione di soglia che fornisce -1 o 1 come valore di output, confrontando la somma pesata degli input rispetto alla soglia. La funzione di output è solo l'identità. Il comportamento della Hopfield Network può dipendere dall'ordine di calcolo come per qualsiasi altra NN Ricorrente, se gli aggiornamenti vengono eseguiti in modo sincrono, significa che i neuroni stanno calcolando contemporaneamente, non potremmo mai raggiungere uno stato stazionario, se invece utilizzando il metodo asincrono, il teorema di convergenza dimostra che troveremo uno stato stazionario al massimo in n*2^n

aggiornamenti. Il vincolo è che dobbiamo usare l'aggiornamento asincrono e che l'ordine di calcolo deve essere fisso, arbitrario ma fisso nel tempo. La dimostrazione del teorema si basa sulla cosiddetta "Funzione Energia". Questo NN e altri NN come le macchine boltzman e le macchine Boltzman con restrizioni si basano sulla definizione di una funzione energia.
Poiché Hopfield Network è composto da neuroni completamente collegati tra loro e i pesi sono simmetrici, esiste un modo molto più semplice per rappresentarli, il grafo di stato.
Ogni nodo rappresenta uno stato specifico, che è composto dalle attivazioni dei vari neuroni, che è fondamentalmente una stringa binaria poiché la funzione di attivazione dei neuroni dà -1 o 1 (potremmo pensare di mappare -1 a 0). Le connessioni rappresentano lo stato conseguente che si raggiunge se aggiorniamo il neurone che viene specificato come peso del bordo.
Possiamo definire l'energia di ogni stato che è fondamentalmente una funzione basata sulla fisica, sta tenendo conto della coerenza di ogni stato e sta definendo un'energia potenziale considerando ogni elemento dello stato (quindi ogni neurone che potrebbe essere -1 o + 1) come una bacchetta magnetica a due poli, se i magneti sono posizionati nello stesso modo avremo un potenziale basso, se sono tutti uno di fronte all'altro significa che abbiamo molta incoerenza e quindi abbiamo un'alta energia potenziale. La natura tende a passare da uno stato di alta energia potenziale a uno stato di bassa energia potenziale, quindi se posizioniamo gli stati del grafico ordinati in base alla loro funzione energetica vedremo che è proprio possibile passare da uno stato con una certa energia a uno stato con un'energia inferiore. Questo è il concetto base del teorema di convergenza.
Un'applicazione delle reti Hopfield è la memorizzazione di modelli specifici. L'idea è di assegnare a ogni stato stazionario uno schema che vogliamo memorizzare. Si può dimostrare che non solo se presentiamo alla rete il pattern, questa si sposterà modificando le attivazioni ma poi raggiungerà quello stato stazionario, ma anche se presentiamo un pattern simile a quello memorizzato dalla rete la capacità di ripristinare il pattern originale, per arrivare così allo stato stady che rappresenta il pattern memorizzato più simile.
E' infatti possibile memorizzare anche più di un modello specifico.

L'idea è di trovare i pesi che avranno raggiunto lo stato costante dello schema che vogliamo memorizzare. Questo viene fatto usando la regola Hebbian Learning. Fondamentalmente rappresentiamo lo schema sulla rete, quindi sui vari neuroni, e poi calcoliamo i pesi che porteranno a quello specifico stato stazionario. Questo si fa assegnando i pesi in questo modo: una volta che abbiamo assegnato il pattern alla rete e quindi ogni neurone ha il valore di un elemento del pattern, diamo il peso 0 alle connessioni dei neuroni con se stessi, il valore 1 se i due neuroni hanno lo stesso valore di attivazione (-1 o 1) e -1 se hanno un valore opposto. Il concetto alla base della regola di apprendimento hebbiano si basa sulla funzione energetica, aumentiamo il peso dei neuroni che hanno una consistenza tra loro e abbassiamo il peso tra i neuroni non coerenti.
La regola può essere estesa anche per memorizzare più di un pattern, l'idea è di valutare la coerenza tra i neuroni sui vari pattern e prenderne una composizione.
Questo approccio e le reti Hopfield potrebbero essere utilizzati anche per risolvere problemi di ottimizzazione. Usiamo un HN per ottimizzare una funzione obiettivo che deve essere minimizzata (ovviamente se dobbiamo massimizzarla, prendiamo il negativo), dopo averla espressa in termini di Funzione Energia. Costruiamo l'HN che rappresenta il problema e utilizziamo come funzione energetica quella che abbiamo appena creato e ogni volta che raggiungiamo uno stato stazionario sappiamo di aver raggiunto almeno un minimo locale

della funzione energetica. Non potrebbe essere il minimo globale perché sappiamo che l'energia tende ad essere più bassa, quindi una volta raggiunto uno stato di bassa energia non c'è modo di uscirne, quindi se il globale non è quello che abbiamo trovato, non lo faremo mai Trovalo.
La simulated annealing è una soluzione a questo problema. La simulated annealing deriva anche dalla fisica ed è fondamentalmente un principio che introduce un termine di temperatura. Col passare del tempo la temperatura si abbassa. Quando la temperatura è alta siamo più liberi di esplorare, questo significa che non ci sposteremo solo verso un minimo ma anche un cambiamento che porta a uno stato energetico più elevato viene accettato con una probabilità specifica, poiché la temperatura si abbassa la probabilità viene portata a 0, quindi a un certo punto cercheremo l'ottimo. Questo risolve il problema di rimanere bloccati nei minimi locali.