Indice

Domande

Reti neurali

- Introduzione
    - Una rete neurale è un grafo diretto G i cui vertici U sono chiamati neuroni o unità e i cui spigoli o archi sono chiamati connessioni. L'insieme dei neuroni è suddiviso in neuroni di input, di output e nascosti. Ciascun neurone inoltre possiede 3 variabili di stato inizializzate da 3 funzioni associate, cioè network input, activation e output. I neuroni di input inoltre hanno una quarta variabile di stato, cioè l'input esterno. Le connessioni invece sono definite da un peso
    - Le reti neurali si dividono in feed-forward network, se il grafo è aciclico, e recurrent network se invece presenta cicli
    - Le operazioni definite su una rete neurale sono la fase di input, in cui l'input esterno è acquisito, e la fase di lavoro, dove si calcola l'output
    - L'apprendimento può essere supervisionato o non supervisionato. Nel primo caso viene fornito un insieme di training pattern, cioè un insieme di vettori di input x con il rispettivo output y, e si conclude quando la rete calcola il medesimo output (potrebbe non essere preciso se finisce il numero massimo di computazioni). Per calcolare l'errore si somma la differenza al quadrato tra l'output atteso e quello calcolato (y - o). Nel secondo caso invece viene sempre fornito un insieme di training pattern formato solamente da un vettore di input x. Non essendoci un output atteso, l'apprendimento si conclude quando input simili hanno output simili. Potrebbe essere necessario normalizzare l'input (val - min / max - min per avere il valore tra 0 e 1).
    - Una rete MLP (MultiLayer Perceptron) è una rete feed-forward con una precisa struttura a strati. La funzione di input per gli strati nascosti e di output è la somma pesata dei suoi input mentre la funzione di attivazione per gli strati nascosti è una funzione sigmoide (monotona non decrescente) e per lo strato di output è una funzione sigmoide o lineare. Funzioni molto usate sono quella logistica, tanh e ReLU (max(0, x) dove ogni valore negativo è messo a 0, molto usata nelle CNN)
- Reti feed forward
    - Nelle reti feed-forward, poiché il grafo non può presentare cicli, la computazione procede dai neuroni di input e progressivamente prosegue fino ai neuroni di output seguendo l'ordine topologico della rete. L'output prodotto rimane costante fintanto che l'output esterno non è prodotto
- 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.
    - Una rete MLP è definita da una rigida suddivisione a strati, quindi è impossibile saltare uno o più livelli.

    - In una rete feed-forward generale, essendoci solo il vincolo di non avere cicli, è possibile costruire delle reti che saltano livelli e sono definite residual neural networks
- In una rete con due neuroni nello stesso livello, l'uscita del primo può essere l'entrata del secondo e viceversa?
    - Non può esistere una rete di tipo feed-forward con questa architettura in quanto si verrebbero a creare dei cicli. Invece per le reti ricorrenti 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 si può usare se si vuole usare discesa del gradiente con error backpropagation.
    - Una funzione sigmoide, sebbene sia anch'essa una funzione non decrescente, è una funzione non lineare, come quella logistica ($1/1+e^{-x}$) o la tangente iperbolica ($e^{x}-e^{-x}/e^{x}+e^{-x}$), e si può calcolare facilmente la derivata tendenzialmente a campana ($f(x)*(1-f(x))$ e quindi molto utile per la discesa del gradiente con error backpropagation

Fuzzy logic

- Introduzione
    - Un fuzzy set μ di un insieme arbitrario, chiamato l'universo, è un mapping $\mu: X \rightarrow [0, 1]$ che assegna a ciascun elemento di X un grado di appartenenza μ(x) all'insieme fuzzy stesso
- Membership function, cosa sono e quali sono le differenze tra i vari tipi?
    - Una membership function quindi esprime l'appartenenza di un elemento all'insieme fuzzy. Per esempio un valore di 1 identifica piena appartenenza mentre un valore di 0 indica assoluta non appartenenza. Con un valore tra 0 e 1 è detta appartenenza parziale.
    - Il significato del grado dipende fortemente dal contesto ma in generale può esprimere il grado di prossimità dell'elemento da degli elementi predefiniti (utile per i cluster per esempio o per i pattern) oppure il grado di preferenza, cioè l'intensità di preferenza di un certo elemento e la fattibilità di poterlo selezionare (utile per analisi decisionali) o infine come grado di possibilità (utile nelle intelligenze artificiali). E' importante, dato il contesto e il significato, scegliere la forma appropriata per la funzione. Triangolare (fuzzy number), a campana, sigmoide, trapezoide (fuzzy interval), ecc….
    - Data una membership function si possono definire alcune caratteristiche particolari come il core, cioè l'insieme di tutti i valori il cui grado di appartenenza è massimo, e il support, cioè l'insieme di tutti i valori il cui grado di appartenenza non è 0. Si possono definire anche insiemi α-cuts, cioè l'insieme degli elementi il cui grado di appartenenza è > α. Per un fuzzy set è possibile anche definire l'altezza, cioè il grado di appartenenza più alto che definisce poi se l'insieme è normale (altezza = 1), o subnormal

- Operazioni in Fuzzy Logic (unione, intersezione e complementare)
    - 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 $\neg 0 = 1$ e $\neg 1 = 0$. 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)

Algoritmi evolutivi

- Introduzione
    - Un algoritmo evolutivo è una ottupla basata su un problema di ottimizzazione ed è definita da G (cioè la popolazione dove l'individuo è definito dalla sua codifica, le informazioni addizionali Z e dal suo valore di fitness), dalla funzione di decodifica dec ($G \rightarrow \mathbf{\Omega}$), dall'operatore di mutazione mut ($G \times Z \rightarrow G \times Z$), dall'operatore di ricombinazione rek ($((G \times Z)^{\text{genitori}} \rightarrow (G \times Z)^{\text{figli}})$), dalle funzioni is_parent e is_environment e da due valori che identificano il numero di genitori che deve avere a popolazione (μ) e il numero di discendenze per generazione (λ)
    - Un problema di ottimizzazione è una tripla data da uno spazio di ricerca ($\mathbf{\Omega}$), una funzione di valutazione ($f: \mathbf{\Omega} \rightarrow R$) che assegna un valore di qualità alle soluzioni candidate e una relazione di comparazione (<, >). L'insieme delle soluzioni è dato da tutti quei valori nello spazio di ricerca il cui valore di qualità è massimo
    - Fattori importanti per un algoritmo evolutivo sono l'encoding di una soluzione, la selezione dei candidati e gli operatori genetici.
      L'encoding non è univoca ma cambia da problema a problema però dovrebbe avere certe caratteristiche, come la rappresentazione di simili fenotipi, cioè soluzioni, in simili genotipi, cioè la loro codifica, e la chiusura sullo spazio di ricerca quando si usano operatori genetici, cioè creando nuove soluzioni candidate esse devono appartenere ancora allo spazio o potrebbero non essere rappresentabili.
      Altro fattore determinante è la selezione. E' spesso collegata al valore di fitness di una soluzione candidata. Con la selective pressure i nuovi individui

vengono selezionati in base al loro valore di fitness e a una variabile temporanea che consente all'inizio di esplorare lo spazio e poi di focalizzarsi intorno agli individui migliori. Ci sono diversi metodi di selezione. Il più usato è sicuramente la roulette-wheel che calcola la probabilità di essere selezionato di ciascun individuo in base al loro valore di fitness. Ha diversi svantaggi quali la dominanza degli individui migliori e la perdita della diversità della popolazione. Da preferirsi quando si vuole ottimizzare un individuo, quindi nelle fasi finali. Un'alternativa può essere il rank based, cioè vengono ordinati gli individui in base al loro valore di fitness e a ciascuno viene assegnato un rank e una distribuzione di probabilità sullo stesso. Successivamente viene utilizzata la roulette wheel. Tuttavia ha un costo di ordinamento (minimo n*log|n|) però evita il problema della dominanza. Un altro metodo può essere il torneo invece. Si prendono k individui e si inseriscono in un torneo dove solo uno vincerà e passerà alla generazione successiva. Facilmente parallelizzabile (visto che il vincitore può essere richiamato) ed evita la dominanza. Altri metodi prevedono di scegliere solamente i migliori nella popolazione oppure tra genitori e figli.

- Operatori genetici negli algoritmi evolutivi
    - Un altro aspetto fondamentale sono gli operatori genetici. Sono di 2 tipi. Mutazione e crossover. L'operatore di mutazione prevede un singolo genitore e prevede di scambiare o sostituire uno o più allele in un gene o uno o più geni in un cromosoma. A seconda di una probabilità di mutazione l'individuo può o non può essere mutato oppure si può utilizzare una gaussiana usando la deviazione standard (se bassa permette di focalizzarsi, se alta permette l'esplorazione). La mutazione binaria comunque è tendenzialmente più veloce e trova facilmente individui interessanti nello spazio.
    - I crossover consistono nello scambiare una o più sequenze di geni a seconda di quanti parti è stato diviso il cromosoma. Si identificano one-point crossover, dove si scambiano tutti i successivi dopo una certa linea, two-point crossover, dove si scambiano i geni tra le due linee, e n-point crossover, dove si alterna tenere e scambiare. Ci sono varianti che prevedono di considerare una certa probabilità di scambio oppure delle permutazioni prima di scambiare. In alternativa si può considerare il cromosoma come un anello o una catena e si scambiano i bordi preservando così l'informazione del vicinato. Nel caso ci siano più di 2 genitori si usa il diagonal crossover, cioè un k - 1 crossover, che scambia le sequenze in modo diagonale.
    - Esistono anche gli operatori di mutazione locali che consiste nell'inversione di una sequenza oppure nel triple exchange e vengono utilizzati in relazione al valore di fitness per migliorare un candidato e raggiungere un massimo locale

- MLP: definition, training e impieghi (NN 4-5-7)
    - Un multilayer perceptron è una rete neurale, cioè un grafo formato da vertici, chiamati neuroni, e lati, chiamati connessioni, di tipo feed forward, cioè è un grafo aciclico, con una struttura ben definita.
    - Una rete neurale è formata da 3 strati: input, hidden e output. Ciascun neurone di ogni strato possiede 3+1 variabili di stato e 3 funzioni. Le variabili

sono network input, activation e l'output e l'input esterno (solo per i neuroni del primo strato). Le funzioni invece sono la funzione di input della rete, la funzione di attivazione e la funzione di output.
    - La funzione di input per lo strato nascosto e di output è la somma pesata degli input dello strato precedente.
    - La funzione di attivazione per lo strato nascosto è una funzione sigmoide, cioè una funzione monotona crescente. Invece per lo strato di output è una funzione lineare o sigmoide.
    - Il training di un MLP si basa sulla minimizzazione dell'errore che può avvenire con più tecniche. Una delle più efficaci è la discesa del gradiente che consente, tramite piccoli step, di migliorare il risultato. Se è usata una funzione di attivazione logistica, il cambiamento dei pesi è proporzionale alla sua derivata (cambia di molto più vicino a 0 aumentando la velocità di apprendimento). Un'altra tecnica che fa uso della discesa del gradiente è la backpropagation. Varianti sono il manhattan training, flat spot elimination, momentum term, self-adaptive error backpropagation, resilient error backpropagation, quick propagation, weight decay.
    - Un MLP può essere usato come funzione di approssimazione per funzioni integrabili, con 4 strati, o funzioni continue, 3 strati. Vengono chiamati approssimatori universali.
- Self-Organizing Maps: definition, configuration (training) algorithms, operations
- Fuzzy sets: definizione, membership function, operazioni (FS 1)
    - Un fuzzy set μ di un insieme X è un mapping $\mu : X \rightarrow [0, 1]$ che assegna ad ogni elemento $x \in X$ un grado di appartenenza $\mu(x)$ al fuzzy set stesso.
    - Una membership function assegna un grado di appartenenza a ciascun elemento del set. Un grado di appartenenza = 1 identifica una piena appartenenza mentre = 0 l'assoluta non appartenenza. La funzione dipende comunque dal contesto e i valori sono identificati solo per convenzione.
- Evolutionary Algorithms: definizione, operatori (EA 1-3)
    - Un algoritmo evolutivo è un problema di ottimizzazione, dato da uno spazio di ricerca,una funzione di valutazione dallo spazio in R e da una funzione di comparazione, ed è una 8-tupla definita da:
        - G-encoded problema di ottimizzazione, funzione di decodifica dal genotipo al fenotipo, funzione di mutazione, funzione di ricombinazione, funzione di discendenza (is_parent), funzione di appartenenza (is_environment), numero di genitori e numero di discendenze per generazione)
    - Gli operatori genetici sono applicati su una parte della popolazione e sono di 3 tipi:
        - 1 genitore: operatore di mutazione che può essere standard, cioè scambio di un allele con uno casuale, oppure a coppia, cioè scambio di un allele con un altro dello stesso cromosoma. Cambia di poco il valore di fitness e può essere applicato casualmente, per l'esplorazione dello spazio, oppure localmente, per migliorare una soluzione candidata

        - 2 genitori: operatore di ricombinazione che può essere one-point crossover, cioè si scambiano tutti i geni al di là di una certa linea, two-points crossover, cioè si scambiando tutti i geni all'interno delle due linee, n-point crossover, cioè si scambiano alternando tenere e scambiare tra le linee, uniform crossover, cioè per ogni gene si sceglie se tenerlo o meno. Ci sono poi altre varianti che possono mischiare i geni, oppure ordinarli, ecc….
        - n-genitori: si applica n-1 crossover spostando diagonalmente i geni
    - L'operatore di selezione non altera in alcun modo gli individui ma ne sceglie una parte in base al loro valore di fitness. Uno dei metodi più usati è la roulette wheel che assegna ad ogni individuo una probabilità di essere selezionato in rapporto alla popolazione. Tuttavia questo metodo si concentra localmente e potrebbe rimanere bloccato in un massimo locale quindi andrebbe utilizzato solo verso la fine.
        - Altri metodi invece sono il rank based, che ordina la popolazione secondo il valore di fitness, l'elitismo che seleziona soltanto i migliori individui, il torneo, che prende un numero ristretto di individui e chi vince potrà passare alla generazione successiva, e il crowding, che prevede la generazione di molti figli e la scelta dei migliori per rimpiazzare i genitori.

Domande generiche

Generiche

- Una rete è un grafo diretto G(U, C) i cui vertici U sono chiamati neuroni o unità e i cui lati C sono chiamati connessioni. L'insieme dei neuroni è suddiviso in 3 categorie: neuroni di input, hidden e di output. Ciascun neurone ha 3 funzioni che generano 3 variabili di stato: network input, activation e output. Inoltre i neuroni di input hanno una quarta variabile di stato, cioè l'input esterno. Ogni connessione tra 2 neuroni possiede un certo peso w.
- Le reti neurali possono essere di due tipi:
    - feed forward: il grafo della rete è aciclico. La computazione procede dai neuroni di input e progressivamente si dirige verso i neuroni di output seguendo l'ordine topologico dei neuroni nella rete. Durante la fase di input, gli input esterni sono acquisiti dai neuroni di input e successivamente sono congelati fintanto che la fase di lavoro non termina. Durante questa fase ogni neurone calcola il proprio output che mantiene costante fintanto che la computazione non termina. Ciò accade quando si genera un output esterno costante oppure quando si arriva al numero massimo di computazioni
    - recurrent: il grafo della rete presenta cicli
- L'apprendimento può essere di due tipi:
    - supervisionato: viene definito un task di apprendimento dato un insieme di neuroni di input, output e un insieme di training pattern, ognuno formato da un vettore di input e vettore di output. L'apprendimento è concluso quando la rete riesce a generare il vettore di output dato il vettore di input

corrispondente. Se ciò non avviene si calcola l'errore sommando ciascun errore per ogni pattern, cioè (output_atteso - output_calcolato)^2.
    - non supervisionato: viene definito un task di apprendimento dato da un insieme di neuroni di input e un insieme di training pattern, ognuno formato da un vettore di input. Non c'è un output desiderato ma input simili dovrebbero avere output simili.
- L'algoritmo c-means è un algoritmo di clustering
    - si sceglie un numero c indicante il numero di cluster desiderati
    - si inizializzano i centri dei cluster in modo casuale
    - si assegna ciascun punto al cluster più vicino
    - si aggiornano i centri (considerando la media dei punti assegnati a ciascun cluster)
    - si ripete il processo di assegnamento e aggiornamento finché i cluster non cambiando

Tipi di rete neurale

- MLP (MultiLayer Perceptron)
    - tipologia: feed-forward con una precisa struttura a strati
    -

Funzione	Input Neurons	Hidden Neurons	Output Neurons
network input		somma pesata dei suoi input	somma pesata dei suoi input
activation		sigmoid function	sigmoid function o lineare
output			

    - Sigmoid function : funzione monotona non decrescente (se bipolare va da -1 a 1 e non da 0 a 1)
        - step, semi lineare, seno fino a saturazione, logistica, tangente iperbolica
    - utilizzi:
        - approssimatore universale
            - ogni funzione integrabile di Riemann è approssimabile con precisione arbitraria da un MLP a 4 strati
            - ogni funzione continua è approssimabile con precisione arbitraria da un MLP a 3 strati
        - deep learning
    - training:
        - supervisionato
        - 2 strati: logistic regression e cambio dei pesi
        - n strati: discesa del gradiente e error backpropagation e cambio dei pesi
    - problemi:

        - ci si può bloccare in un minimo locale. Bisogna ripetere il processo con differenti punti iniziali e poi si sceglie la soluzione migliore
        - underfitting: se il numero dei neuroni nello strato nascosto è troppo basso, si potrebbe perdere qualche relazione tra input e output
        - overfitting: se il numero dei neuroni nello strato nascosto è troppo alto, si potrebbe adattare troppo e potrebbe essere influenzato da errori o outliers del training set
        - i risultati potrebbero essere difficili da interpretare data la loro natura codificata in una matrice
- Deep Learning (da MLP)
    - utilizzi:
        - auto-encoder 3 strati
            - mappa i suoi input con la loro approssimazione con hidden layer avente funzione di encoder e output layer con funzione di decoder
        - stacked auto-encoder 4 strati
            - mappa i suoi input con la loro approssimazione con primo hidden layer che costruisce le caratteristiche primarie e secondo hidden layer che costruisce le caratteristiche secondarie per la ricostruzione
        - CNN (Convolutional Neural Networks)
            - ispirate alla retina umana, i neuroni di input sono collegati a una regione limitata e ogni neurone nascosto del primo layer è collegato a un numero ristretto di neuroni dell'input layer. Lo spazio di input è mosso sull'intera immagine creando una convoluzione con un piccolo kernel
            - ogni strato successivo elabora caratteristiche sempre più ad alto livello (partendo dai bordi arrivando a un'immagine)
    - problemi:
        - overfitting: si potrebbero avere troppi neuroni nascosti dato da un numero maggiore di parametri variabili → evitabile avendo un decadimento dei pesi e quindi non pesi troppo grandi e limitando il numero dei neuroni nascosti e anche il numero di quelli attivi
        - scomparsa del gradiente: il gradiente tende a scomparire avendo troppi strati e l'apprendimento in quelli iniziali viene quindi rallentato → evitabile cambiando funzione di attivazione o aumentando i pesi
- RBF (Radial Basis Function)
    - tipologia: feed-forward a 3 strati
    -

Funzione	Input Neurons	Hidden Neurons	Output Neurons
network input		funzione di distanza tra il vettore di input e quello dei pesi	somma pesata degli input

activation		radial basis function	funzione lineare
output			

    - Radial basis function: funzione monotona decrescente con una regione definita da un raggio di riferimento σ
        - rettangolo, triangolo, coseno fino a 0, gaussiana
    - Distance function: Minkowski family $(\sum_{i=1}^{n} |x_i - y_i|^k)^{\frac{1}{k}}$
        - manhattan, euclidea, ecc…
    - utilizzi:
        - approssimatore di funzioni
    - training:
        - apprendimento supervisionato
        - semplice
            - un neurone nascosto per ogni training pattern
            - se la funzione di attivazione è gaussiana → raggio scelto euristicamente
            - inizializzazione delle connessione tra neuroni nascosti a neuroni di output (matrice output ottenuti * matrice pesi = vettore output desiderato)
            - si ricava la matrice dei pesi invertendo la matrice degli output ottenuti moltiplicata gli output desiderati
        - generale
            - si seleziona un subset dei training pattern come centri con i training pattern come peso per i neuroni nascosti e la matrice degli output ottenuti come peso per i neuroni di output
            - data la matrice per i neuroni di output, si inverte e si trovano i pesi
        - l'aggiornamento prevede di utilizzare il gradiente per aggiornare i pesi dai neuroni nascosti a quelli di output, i pesi dai neuroni di input a quelli nascosti e il raggio delle radial basis functions. Si può inoltre usare la tecnica della backpropagation per aggiornare i parametri della rete
    - problemi:
        - scelta dei centri nell'approccio generale
            - tutti i dati come centri (raggio e pesi da calcolare ma si possono ottenere risultati precisi)
            - random subset (veloce ma raggio e pesi da calcolare e performance variabili)
            - clustering (c-means o LVQ)
- LVQ (Learning Vector Quantization)
    - tipologia: feed-forward a 2 strati (RBF con hidden layer = output layer)
    -

Funzione	**Input Neurons**	**Hidden Neurons**	**Output Neurons**

network input		X	funzione di distanza tra il vettore di input e il vettore dei pesi
activation		X	radial basis function
output		X	considera l'attivazione di tutti i neuroni

    - Distance function: Minkowski family $(\sum_{i=1}^{n} |x_i - y_i|^k)^{\frac{1}{k}}$
        - manhattan, euclidea, ecc…
    - Radial basis function: funzione monotona decrescente con una regione definita da un raggio di riferimento σ
        - rettangolo, triangolo, coseno fino a 0, gaussiana
    - utilizzi:
        - classificatore
    - training:
        - apprendimento non supervisionato
        - per ogni training pattern trovare il più vicino vettore di riferimento (calcolando la distanza)
        - adatta soltanto il vettore di riferimento del neurone vincente (o i due più vicini per dati classificati)
        - ogni vettore è assegnata a una classe
        - regole di aggiornamento:
            - attraction rule: se il punto e il vettore hanno la stessa classe, si avvicinano
            - repulsion rule: se il punto e il vettore hanno una classe diversa, si allontanano
    - problemi:
        - un learning rate fisso può portare a oscillazioni → eliminabile avendo un time dependent learning rate
        - i vettori di riferimento potrebbero allontanarsi troppo → eliminabile applicando una finestra di aggiornamento, cioè l'aggiornamento si ferma se è abbastanza lontano
- SOM (Self Organizing Map o Kohonen feature map)
    - tipologia: feed-forward a 2 strati
        - simile a LVQ solo che che le connessioni sono soltanto con i neuroni vicini
    -

Funzione	**Input Neurons**	**Hidden Neurons**	**Output Neurons**
network input		X	funzione di distanza tra il vettore di input e il vettore dei pesi
activation		X	radial basis function

output		X	funzione identità

- Distance function: Minkowski family $(\sum_{i=1}^{n} |x_i - y_i)^k)^{\frac{1}{k}}$
  - manhattan, euclidea, ecc…
- Radial basis function: funzione monotona decrescente con una regione definita da un raggio di riferimento σ
  - rettangolo, triangolo, coseno fino a 0, gaussiana
- L'output è spesso discretizzato secondo il principio "winner takes all" e per ciascun neurone di output è definita una relazione di vicinanza
  - una griglia quadrata o esagonale
  - la funzione di vicinanza è una funzione radiale e utilizza un time-dependent learning rate e un time-dependent neighborhood radius, quindi la grandezza del vicinato si riduce nel tempo
- utilizzi:
  - preserva la topologia della rete ma le distanze, angoli e aree possono essere distorti
  - riduzione delle dimensioni
- training:
  - apprendimento non supervisionato
  - inizializzazione del vettore dei pesi con vettori di riferimento casuali
  - scelta di un esempio di apprendimento
  - trovare il neurone vincente (quello più vicino al vettore di riferimento)
  - calcola il raggio e learning rate e adatta i vicini al neurone vincente
- problemi:
  - l'apprendimento può fallire se il learning rate iniziale è troppo basso oppure il raggio iniziale è troppo basso

- Recurrent
  - grafo presenta cicli e l'output è generato quando si raggiunge uno stato stabile
  - error backpropagation considera una variabile temporale per evitare loop infiniti
  - vectorial neural networks: insieme di recurrent networks
    - utilizzate per calcolare equazioni vettoriali
- Hopfield
  - tipologia: recurrent a 2 strati
    - ogni neurone di input è un neurone di output e viceversa
    - ogni neurone è collegato con tutti gli altri eccetto se stesso e i pesi delle connessioni sono simmetrici
  -

Funzione	**Input Neurons**	**Hidden Neurons**	**Output Neurons**
network input	somma pesata degli output di tutti gli altri neuroni	X	somma pesata degli output di tutti gli altri neuroni

activation	funzione di soglia	X	funzione di soglia
output	funzione identità	X	funzione identità

    - Threshold function: funzione il cui output è 1 se l'input è maggiore della soglia, altrimenti -1 (attrazione o repulsione)
    - matrice dei pesi:
        - 0 sulla diagonale maggiore
        - peso della connessione tra i due neuroni altrimenti
    - aggiornamento:
        - sincrono: in parallelo → la computazione può oscillare a seconda dell'ordine di aggiornamento
        - asincrono: sequenziale → la computazione converge indipendentemente dall'ordine di aggiornamento (se vengono aggiornati ciclicamente in un ordine qualsiasi ma fisso in al più $n*2^n$ passi per il teorema di convergenza - si prova tramite l'ausilio di una funzione di energia) → si raggiunge uno stato stabile
    - utilizzi:
        - ricostruzione di dati corrotti tramite memoria associativa
        - rappresentazione della memoria umana
        - ottimizzazione di funzione
    - training (ottimizzazione):
        - trasformazione della funzione da ottimizzare in una funzione da minimizzare e poi in una funzione di energia per una rete di Hopfield
        - estrapola pesi e valori di soglia dalla funzione
        - costruisci la rete di Hopfield corrispondente
        - inizializzazione casuale della rete e aggiornamento fino a convergenza
        - leggi la soluzione dallo stato stabile raggiunto
        - ripeti il processo
    - Hebbian learning rule
        - si usano gli stati stabili per salvare i pattern → memoria associativa
        - si inizia salvando un singolo pattern, cioè si trovano i pesi per raggiungere uno stato stabile
        - successivamente si salvano tutti gli altri pattern
    - problemi:
        - la computazione può oscillare se i neuroni sono aggiornati in parallelo
        - bloccato in un massimo locale → simulated annealing, cioè le transizione da un minimo più alto a un minimo più basso sono più probabili → una soluzione migliore è sempre accettata mentre una peggiore dipende dalla qualità della soluzione e dalla temperatura
- Boltzmann
    - tipologia: recurrent a 2 strati
        - simili alle reti di Hopfield ma cambia come vengono aggiornati i neuroni e la possibilità di assegnare energia ai neuroni e possiede neuroni nascosti
    -

Funzione	Input Neurons	Hidden Neurons	Output Neurons
network input	somma pesata degli output di tutti gli altri neuroni	somma pesata degli output di tutti gli altri neuroni	somma pesata degli output di tutti gli altri neuroni
activation	funzione logistica	funzione logistica	funzione logistica
output	funzione identità	funzione identità	funzione identità

- La probabilità di un neurone di essere attivo è una funzione logistica basata sulla differenza di energia tra il suo stato attivo e il suo stato inattivo
- utilizzi:
  - risoluzione di problemi combinatorici
- aggiornamento macchina di Boltzmann:
  - un neurone è scelto casualmente
  - viene calcolata la differenza di energia e la probabilità di attivazione
  - al neurone viene assegnata questa probabilità per attivazione 1 e 1-p per attivazione 0
  - si ripete il processo più volte per neuroni casuali
  - alla fine la probabilità che la rete sia in uno specifico stato di attivazione dipende solamente dall'energia di quello stato → equilibrio termico
- training machine
  - lo scopo è adattare la distribuzione di probabilità rappresentata da una Boltzmann machine a un insieme di dati
  - positive phase → i neuroni visibili sono aggiornati rispetto a un punto casuale e i neuroni nascosti sono aggiornati fino al raggiungimento dell'equilibrio termico
  - negative phase → tutti i neuroni sono aggiornati fino al raggiungimento dell'equilibrio termico e lo stato non è determinato da dati esterni
- problemi:
  - bloccato su un massimo locale → simulated annealing implementato abbassando lentamente una variabile di temperatura
  - il raggiungimento dell'equilibrio termico diventa sempre più lungo tanto più diventa grande la macchina
- variante restricted
  - grafo bipartito e i neuroni divisi in visibili e nascosti
  - le connessioni ci sono solo tra neuroni di gruppi diversi
  - tutti i neuroni di input sono neuroni di output e i neuroni nascosti sono diversi per input e output
  - training
    - le unità visibili sono aggiornate per un certo input
    - le unità nascoste sono aggiornate una volta e in parallelo
    - neuroni nascosti e visibili sono fissati e aggiornati nuovamente e si calcola il gradiente negativo

            - si aggiornano i pesi considerando la differenza tra il gradiente positivo e negativo
    - variante deep
        - utilizzo della variante restricted per costruire deep neural networks
        - si costruiscono 2 reti restricted, una sull'input e una sulle attivazioni dei neuroni nascosti
        - si aggiorna tramite backpropagation

Fuzzy set

- Un fuzzy set μ di un insieme X, chiamato universo, è un mapping μ : X -> [0, 1] che assegna ad ogni elemento x ∈ X un grado di appartenenza μ(x) all'insieme fuzzy stesso.
- La funzione di appartenenza assegna un grado di appartenenza a ciascun elemento. Un grado = 1 rispecchia una piena appartenenza mentre 0 la totale non appartenenza. Un crisp set è un caso particolare di fuzzy set dove gli unici valori di appartenenza consentiti sono 0 e 1.
- I gradi di appartenenza dipendono dal contesto e non sono fissati a priori. Essi rappresentano inoltre 3 significati diversi:
    - somiglianza, cioè il grado di prossimità di un elemento da altri elementi prototipo. Importante per classificazione di pattern e cluster analysis
    - preferenza, cioè l'intensità di preferenza in favore di un oggetto e la fattibilità di selezionarlo. Importante per fuzzy optimization e decision analysis
    - possibilità, cioè la possibilità che un parametro X abbia un certo valore. Importante per intelligenze artificiali
- Per un fuzzy set si può definire un α cut, cioè un sottoinsieme di X tale che il grado di appartenenza di ciascun elemento è > o ≥ di un certo valore α. Altri insiemi particolari definibili per un fuzzy set sono il support, cioè un sottoinsieme di X tale che il grado di appartenenza è > 0 e il core, cioè un sottoinsieme di X tale che il grado di appartenenza è 1. E' inoltre definibile l'altezza di un fuzzy set, cioè il grado di appartenenza più alto riscontrabile nell'insieme stesso. Se è presente un elemento con grado = 1, allora l'insieme è detto normal, altrimenti subnormal.
- Gli operatori di un fuzzy set sono complemento ¬ μ (1-μ(x)), intersezione μ ∧ μ' (min(μ(x), μ'(x)) implementata con una t-norm e unione (max(μ(x), μ'(x)) implementata con una conorm (cioè funzioni che hanno un elemento identità, sono commutative e associative)

Fuzzy controller

- I fuzzy control sono l'impiego più diffuso dei fuzzy system. A differenza dei normali sistemi essi mappano l'input analogico in un range [0, 1] e non soltanto 0 e 1. La loro architettura è suddivisa in 4 componenti:
    - knowledge base: è suddiviso in data base che contiene le eventuali trasformazioni del dominio, l'associazione di un termine linguistico con il suo fuzzy set corrispondente e informazioni sui limiti e in rule base, cioè l'insieme di tutte le regole linguistiche di controllo (spesso nella forma IF <variabile> IS

<valore> [AND/OR <altre condizioni>] THEN <variabile> IS <valore> (mamdani) oppure <funzione con valori> (takagi))
    - fuzzification interface: riceve l'input crisp, ma mappa in un altro dominio se necessario secondo delle trasformazioni contenute nel knowledge base, e successivamente lo converte in termini linguistici o fuzzy set
    - decision logic: si occupa di processare i dati e di calcolare un output in base alle regole contenute nel knowledge base (usando regole di inferenza quali modus ponens e modus tollens. Per ogni regola si andrà a calcolare in modo congiuntivo tramite un t-norm o disgiuntivo tramite una co-norm il valore combinato di tutti gli input). Nel caso del Takagi-Sugeno controller, l'output sarà già crisp, altrimenti sarà fuzzy
    - defuzzification interface: determina il valore crisp dato l'output ed eventualmente lo mappa al dominio appropriato tramite:
        - max criterion: si sceglie il valore che porta al massimo grado di appartenenza. Applicabile per qualsiasi fuzzy set o dominio ma i valori sono casuali (approccio non deterministico) e ci potrebbero essere azioni discontinue
        - mean of maxima: se l'output fuzzy è un intervallo, si calcola l'insieme dei valori con il grado di appartenenza più alto e si calcola la media (somma/cardinalità se finito altrimenti integrale per ogni valore di Ymax / integrale per ogni Ymax (int 1)). Ci potrebbero essere azioni discontinue
        - center of gravity: se l'output fuzzy è un intervallo, si calcola il centro dell'area indicata da quell'intervallo (somma per ogni y * output/somma per ogni y dell'output se finito altrimenti integrale). Tendenzialmente ha un comportamento costante ma ha una computazione lunga e risultati controintuitivi

Fuzzy clustering

- L'obiettivo di questo task non supervisionato è quello di dividere il dataset in modo tale che:
    - oggetti il più simile possibile appartengono allo stesso cluster
    - oggetti il più dissimile possibile appartengono a cluster diversi
- La somiglianza è data da una funzione di distanza (minkowski family) dove:
    - $d(x, y) = 0$ sse $x = y$ (identità)
    - $d(x, y) = d(y, x)$ (simmetria)
    - $d(x, y) \leq d(x, y) + d(y, z)$ (disuguaglianza triangolare)
- c-means clustering
    - scegli un numero c di cluster (user input)
    - inizializza i centri dei cluster in modo casuale (scegliendo c punti)
    - assegna i punti al cluster più vicino
    - aggiorna i centri dei cluster (tramite media o centro di gravità)
    - ripetere il processo finché i centri non cambiano

- Il processo di clustering è dipendente dalla scelta iniziale dei centri e di solito converge ma potrebbe rimanere bloccato in un continuo aggiornamento dei centri in caso di massimo locale
    - il clustering è ottimale quando la somma delle distance quadrate tra i centri dei cluster e i punti ad esso assegnati è minima
        - $J_f(X, U_h, C) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} d_{ij}^{2}$ bisogna minimizzare, con m > 1 che indica il fuzzifier (m = 1 hard c-means) e $d_{ij} = d(c_i, x_j)$ (distanza di un punto dal centro)
        - per ottimizzare J_f si utilizza il metodo dei moltiplicatori di lagrange. I parametri di ottimizzazione sono forma e grandezza dei cluster e la funzione di distanza
    - ogni punto appartiene a un solo cluster e ogni cluster ha almeno un punto
    - si ripete il processo e si sceglie il risultato migliore
- Nei fuzzy cluster i punti hanno un grado di appartenenza al cluster tra 0 e 1
    - fuzzy cluster probabilistico FCM (Fuzzy C-Means):
        - $\sum_{j=1}^{n} u_{ij} > 0 \quad \forall i \in [1, c]$, con u_{ij} che indica l'appartenenza del punto x_j al cluster Γ_i, cioè ad ogni cluster è attribuito almeno un punto
        - $\sum_{i=1}^{c} u_{ij} = 1 \ \forall i \in [1, n]$, cioè la somma di tutti i gradi di appartenenza in tutti i cluster è 1
        - il grado di appartenenza dipende da tutti gli altri cluster
    - fuzzy cluster possibilistico PCM (Possibilistic C-Means):
        - $\sum_{j=1}^{n} u_{ij} > 0 \quad \forall i \in [1, c]$, con u_{ij} che indica l'appartenenza del punto x_j al cluster Γ_i, cioè ad ogni cluster è attribuito almeno un punto
        - la somma di tutti i gradi di rappresentatività non deve essere 1
        - J_f da ottimizzare non si può utilizzare in quanto alcuni punti (outlier) non sono assegnati ad alcun cluster, quindi è aggiunto è termine di penalizzazione
        - il grado di appartenenza dipende soltanto dal cluster stesso
    - gustafson-kessel
        - cluster non circolari in quanto cambia la funzione di distanza → Manalanobis distance e non euclidea
    - FCV (Fuzzy C-Varieties)
        - riconosce linee, piani iperpiani
    - kernel based fuzzy clustering
        - modifica della funzione di distanza per gestire dati non vettoriali come alberi e grafi (converte gli algoritmi lineari in non lineari)
        - basato su un mapping dallo spazio di input in uno spazio in n dimensioni, chiamato spazio di Hilbert, in cui i dati sono gestiti dai dot product
        - nel caso di uso di distanza euclidea, identico a FCM
- Per eliminare i noisy data si possono aggiungere dei noisy cluster, i cui centri sono equidistanti da tutti i punti

Neuro fuzzy

- I sistemi neuro-fuzzy combinano i vantaggi della computazione parallela e le abilità di apprendimento delle reti neurali con la capacità di rappresentazione e spiegazione della conoscenza dei sistemi fuzzy → reti neurali semplificate e fuzzy system capaci di apprendere
- Modello cooperativo
    - la rete neurale e il sistema fuzzy lavorano in modo indipendente
    - la rete neurale opera online (ottimizza certi parametri) o offline (genera certi parametri)
- Modello ibrido
    - mappa i fuzzy set e le regole fuzzy in una rete neurale
    - struttura integrata, nessuna comunicazione necessaria tra i due componenti e la rete opera sia offline che online
    - fuzzy set antecedenti le regole:
        - peso delle connessioni → nel primo livello nascosto i neuroni rappresentano gli antecedenti della regola
        - funzione di attivazione → nel primo livello nascosto i neuroni rappresentano i fuzzy set e i neuroni nel secondo livello gli antecedenti della regola
    - algoritmo:
        - ogni variabile di input diventa un neurone nello strato di input
        - ogni variabile di output diventa un neurone nello strato di output
        - per ogni fuzzy set crea un neurone nel primo strato nascosto e connettilo al corrispondente input
        - per ogni regola crea un neurone nel secondo strato nascosto e specifica una t-norm per computare l'attivazione della regola
        - connetti ogni regola al suo fuzzy set antecedente
        - output:
            - mamdani - connetti ogni regola al suo output e usa come peso della connessione il fuzzy set conseguente la regola
            - takagi - per ogni neurone regola crea un fratello che calcola il valore di output per quella regola
    - addestramento con error backpropagation (cercando di minimizzare la somma degli errori quadrati e utilizzando la discesa del gradiente se applicabile)
    - ANFIS (Adaptive Network-based Fuzzy Inference Systems):
        - integra le regole fuzzy in una rete neurale (secondo modello di takagi)
    - NEFCON (NEuro Fuzzy CONtroller)
        - integra le regole fuzzy in una rete neurale (secondo modello mamdani)
    - NEFCLASS (NEuro Fuzzy CLASSification)
        - rappresenta le regole che fanno una predizione sulle classi

Neural Networks

Introduzione

Aspetto biologico

- I neuroni comunicano tra loro tramite gli assoni (axon) che lasciano attraversare le informazioni solo quando sono depolarizzati
    - c'è bisogno di un forte impulso che fa rilasciare dei neurotrasmettitori che si occupano di far cambiare la polarizzazione della membrana

Vantaggi

- Alto velocità di elaborazione dati
    - grazie all'utilizzo massivo del parallelismo
- Fault tolerance
    - anche se una parte della rete neurale non risponde, essa continua a funzionare
- Degradazione costante
    - se una parte della rete non funziona, c'è un calo graduale delle performance (dovuto alla non risposta dei neuroni)
- Adatto all'apprendimento induttivo
    - dato dagli esempi e dalla generalizzazione delle istanze

Struttura di una rete neurale artificiale

Threshold Logic Unit (TLU)

- Sono la base delle reti neurali (corrispondenti ai neuroni) e sono esclusivamente per numeri
- Sono unità logiche di soglia, cioè hanno un output solo se una certa condizione è rispettata (superamento del valore di soglia)
- Struttura:
    - input x_1, …, x_n con ciascuno un peso w_1, …, w_n
    - output
        - 1 se $\sum_{i=1}^{n} x_i w_i \geq \theta$ dove θ indica il valore di soglia
        - 0 altrimenti
- Esempi singola TLU:
    - $x_1 \wedge x_2 = y$
        - x_1 ha peso 3
        - x_2 ha peso 2
        - il valore di soglia è 4

x_1	x_2	$s(3x_1+2x_2)$	$y(s>4)$
0	0	0+0=0	0
0	1	0+2=2	0
1	0	3+0=3	0
1	1	3+2=5	1

    - $x_2 \rightarrow x_1 = y$
        - x_1 ha peso 2
        - x_2 ha peso -2
        - il valore di soglia è -1
        -

x_1	x_2	$s(2x_1-2x_2)$	$y(s>-1)$
0	0	0-0=0	1
0	1	0-2=-2	0
1	0	2-0=2	1
1	1	2-2=0	1

- L'applicazione di una TLU consente di tracciare una linea nello spazio suddividendo i valori veri da quelli falsi
    - a volte è necessario utilizzare più TLU e quindi di tracciare più linee per effettuare questa separazione
- Esempi multiple TLU:
    - $x_1 \leftrightarrow x_2$
        - x_1 ha peso -2 per TLU_1 e 2 per TLU_2
        - x_2 ha peso 2 per TLU_1 e -2 per TLU_1
        - TLU_1 ha peso 2 per TLU_3
        - TLU_2 ha peso 2 per TLU_3
        - il valore di soglia per TLU_1 è -1
        - il valore di soglia per TLU_2 è -1
        - il valore di soglia per TLU_3 è 3
        - $s_1=2x_2-2x_1$ e $y_1=s_1>-1$, $s_2=2x_1-2x_2$ e $y_2 = s_2>-1$, $s_3=2y_1+2y_2$ e $y_3 = s_3>3$
        -

x_1	x_2	s_1	y_1	s_2	y_2	s_3	y_3
0	0	0	1	0	1	4	1
0	1	2	1	-2	0	2	0
1	0	-2	0	2	1	2	0
1	1	0	1	0	1	4	1

Training TLU

- Algoritmo apprendimento automatico:
  - valori random per i pesi e valori di soglia
  - valutazione dell'errore tramite una funzione
  - cambiare i valori dei pesi e dei valori di soglia per ridurre l'errore
  - ripetere il procedimento
- Modalità di apprendimento:
  - online:
    - basato su un pattern di apprendimento per volta
    - calcola e applica i parametri di correzione per il pattern
  - batch:
    - basato su un insieme di pattern di apprendimento
    - calcola e applica i parametri di correzione per tutto l'insieme di pattern
- Delta rule
  - Sia $(x_1, \ldots, x_n)^T$ il vettore degli input per una TLU e sia y il suo output. Dato o come l'output desiderato, se $o \neq y$ allora i pesi e i valori di soglia sono adattati in questo modo per ridurre l'errore $\forall i \in \{1, \ldots, n\}$ con η che indica la curva di apprendimento:
    - $\theta^{new} = \theta^{old} + \Delta\theta$, con $\Delta\theta = -\eta(o-y)$
    - $w_i^{new} = w_i^{old} + \Delta w_i$, con $\Delta w_i = \eta(o-y)x_i$
- Teorema di convergenza
  - Sia L un insieme di pattern di apprendimento (coppie di vettori di input e output desiderati) e sia L_0 il pattern con output desiderato 0 e L_1 il pattern con output desiderato 1. Se L_0 e L_1 sono linearmente separabili, cioè se esiste un vettore dei pesi e un valore di soglia tale che la somma tra input e pesi e minore (per L_0) o maggiore (per L_1) della soglia per ogni possibile input del patter, allora sia l'apprendimento online che quello a batch termineranno
  - per i problemi non linearmente separabili, l'algoritmo non termina
  - utilizzando lo schema ADAptive LINear Element che prevede la codifica di falso con -1, si velocizza la minimizzazione dell'errore

Artificial Neural Networks

- Una rete neurale è un grafo diretto ($G = (U, C)$) i cui vertici sono chiamati neuroni(U) o unità e i lati (C) sono chiamati connessioni
- I neuroni:
  - si suddividono in neuroni di input U_{in}, neuroni di output (U_{out}) e neuroni nascosti (U_{hidden})
  - hanno 3+1 variabili di stato:
    - network input net_u
    - activation act_u
    - output out_u
    - external input ext_u (solo neuroni di input)
  - hanno 3 funzioni:
    - network input function $f^{(u)}_{net}$

        - activation function $f^{(u)}_{act}$
        - output function $f^{(u)}_{out}$
- Le connessioni:
    - hanno un peso w_{uv}
- Le reti neurali si differenziano in:
    - feed-forward network se il suo grafo è aciclico
        - la computazione procede dai neuroni di input a quello di input seguendo l'ordine topologico della rete
        - durante la computazione non si accettano nuovi input e ogni neurone mantiene stabile il suo output fintanto che non viene generato l'output esterno
    - recurrent network se il suo grafo contiene cicli
        - continua attivazione e generazione di output dei neuroni finché non si raggiunge uno stato stabile
- Fasi di una rete neurale:
    - input phase: i neuroni di input acquisiscono gli input esterni
    - work phase: fintanto che l'input viene processato, non si accettano nuovi input esterni
        - se l'input cambia avviene la ricomputazione di un neurone di output
        - la fase termina quando l'output è costante oppure si è raggiunto il numero massimo di ricomputazioni
- Apprendimento:
    - consiste nell'aggiornare il peso delle connessioni ed eventualmente i valori di soglia per ottimizzare un criterio oggettivo:
        - obiettivo di apprendimento fissato (apprendimento supervisionato)
            - insieme di pattern di apprendimento $l = (\vec{i}(l), \vec{o}(l))$ con vettore di input i $(ext_{u1}(l), \ldots, ext_{un}(l))$ e un vettore di output o $(o_{v1}(l), \ldots, o_{vm}(l))$
            - errore: $\sum_{v \in L_{fixed}} (o_v{}^{(l)} - out_v{}^{(l)})^2$ dove o è l'output desiderato e out è l'output ottenuto (L_{fixed} è l'obiettivo e bisogna calcolare l'errore per ogni pattern fornito)
        - obiettivo di apprendimento libero (apprendimento non supervisionato)
            - insieme di pattern di apprendimento $l = (\vec{i}(l))$ con vettore di input i $(ext_{u1}{}^{(l)}, \ldots, ext_{un}{}^{(l)})$
            - non c'è un output desiderato ma input simili dovrebbero avere output simili
            - prima di processare bisogna normalizzare il vettore di input per avere con il valore atteso una media aritmetica pari a 0 e una deviazione standard pari a 1

Percettrone

- Un r-layer perceptron è un feed-forward network con una struttura ben definita
- La funzione di input di ogni neurone nascosto e di output è la somma ponderata dei suoi input (somma di $w_{uv}out_v$ per ogni v)

- La funzione di attivazione di ogni neurone nascosto è la funzione sigmoidea, cioè una funzione monotona crescente
    - quella dei neuroni di output è una funzione sigmoidea o lineare
    - f_{act}(net, θ) = 1 se net ≥ θ, 0 altrimenti (sigmoidea)
    - f_{act}(net, θ) = 1 se net > θ + ½, 0 se net < θ - ½, (net - θ) + ½ altrimenti (semi lineare)
    - f_{act}(net, θ) = 1 se net > θ + π/2, 0 se net < θ - π/2, (sin(net - θ) + 1)/2 altrimenti (seno fino a saturazione)
    - f_{act}(net, θ) = 1/(1+e^(-net+θ))
    - alcune funzioni sigmoidee bipolari (da -1 a 1) come la tangente iperbolica sono usate
- Ogni funzione integrabile di Riemann può essere approssimata con arbitraria precisione da un percettrone a 4 strati
    - ogni funzione continua può essere approssimata da un percettrone a 3 strati
    - il percettrone a multi strato è detto approssimatore universale data la sua topologia e struttura

Regressione

Lineare

- Dato un dataset ((x_1, y_1), …, (x_n, y_n)) e un'ipotesi di relazione funzionale g(x) = a + bx, l'obiettivo è minimizzare la somma dei quadrati degli errori
    - $F(a,\ b) = \sum_{i=1}^{n}(g(x_i) - y_i)^2 = \sum_{i=1}^{n}(a + bx_i - y_i)$
- Soddisfatte le condizioni di minimo del teorema di Fermat ed escluso che tutte le x abbiano il medesimo valore, la soluzione è unica e la linea risultante è detta linea di regressione
    - $\frac{\delta F}{\delta a} = \sum_{i=1}^{n} 2(a + bx_i - y_i) = 0$
    - $\frac{\delta F}{\delta b} = \sum_{i=1}^{n} 2(a + bx_i - y_i)x_i = 0$
- Si può generalizzare con un polinomio di ordine qualsiasi e con funzioni a più argomenti(g(x,y))

Logistic

- Dato un dataset ((x_1, y_1), …, (x_n, y_n)) e una funzione non polinomiale $y = ax^b$, l'obiettivo è trovare una trasformazione lineare/polinomiale lny = lna + b lnx
    - $y = \frac{Y}{1+e^{a+bx}} \Leftrightarrow \frac{1}{y} = \frac{1+e^{a+bx}}{Y} \Leftrightarrow \frac{Y-y}{y} = e^{a+bx}$
    - $ln(\frac{Y-y}{y}) = a + bx$
- Si può generalizzare con funzioni a 2 argomenti
    - si può utilizzare però solo con percettroni a 2 strati

Discesa del gradiente

- Il gradiente è dato dal calcolo dell'errore minimo a piccoli step (identificando la direzione dello step) (?)

- Se si usa la funzione di attivazione logistica, i pesi cambiano in modo proporzionale alla derivata della stessa
    - vicino a 0 in modo significativo aumentando la velocità di apprendimento
    - lontano da zero il gradiente è basso e l'apprendimento è lento
- L'errore, che non potrà mai essere nullo data la natura della funzione logaritmica, verrà trasmesso a tutti i neuroni in modo contrario alla direzione (backpropagation)

- Il numero dei neuroni nascosti può essere:
    - underfitting: il numero è troppo piccolo e il percettrone multistrato potrebbe non cogliere le relazioni tra input e output
    - overfitting: il numero è troppo grande e il percettrone multistrato potrebbe essere influenzato da errori o deviazioni anomale del training set
- Cross validation:
    - dividere il dataset in training e validation dataset
        - dividi in n parti uguali. n-1 per training, 1 per validation
    - allena il percettrone con differenti numeri di neuroni nascosti sul training e testalo sul validation
    - ripeti il processo splittando in modo casuale il dataset e facendo una media
        - se l'errore in test diminuisce troppo, rischia di aumentare quello sul validation → overfitting
    - scegli il numero di neuroni nascosti che il miglior errore medio
- Calcolando quali output differiscono maggiormente da altri output con input simili, si possono escludere dal dataset

Deep learning

- ???
- Svantaggi del deep learning:
    - overfitting: un numero troppo elevato di stati e neuroni
        - effettuare un decadimento del peso permette di non avere pesi troppo alti, quindi un adattamento troppo preciso
        - soltanto un numero ristretto di neuroni sono attivi negli strati nascosti
        - alcuni neuroni possono essere scartati randomicamente durante il test
    - scomparsa del gradiente: il gradiente tende a sparire all'aumentare degli strati e l'apprendimento nei primi strati diventa molto lento
        - un gradiente basso può essere eliminato tramite un peso alto che però satura la funzione di attivazione
- Auto-encoder
    - è un percettrone a 3 strati che mappa un input con la sua approssimazione
    - se gli strati nascosti hanno un numero di neuroni almeno pari all'input, l'approssimazione non è garantita (in quanto la attraversa senza modifiche)
        - basta diminuire il numero di neuroni nascosti per forzare l'apprendimento di caratteristiche e limitare i neuroni che si possono attivare
        - aggiungere del rumore in input può aiutare a forzare l'apprendimento

    - nella versione a stack, il primo strato identifica le caratteristiche primarie (relative al raw input) mentre il secondo strato identifica le caratteristiche secondarie (relative ai dati). Il predittore è allenato dal secondo strato
- Utilizzi (CNN Convolutional Neural Networks):
    - ispirate alla retina umana (c'è un campo limitato che risponde agli stimoli)
    - è una rete neurale feed-forward
    - i neuroni del primo strato sono connessi a una piccola parte dei neuroni in input che si occupa di una regione specifica del campo visivo
    - i successivi strati applicano maximum pooling su piccole regioni che mantiene le informazioni della regione ma non la sua localizzazione nell'immagine
    - buone performance per riconoscimento della grafia e di oggetti in un'immagine

Radial Basis Function Networks

- Una RBF è una rete neurale feed-forward a 3 strati con una radial basis function come funzione di attivazione nello strato nascosto
- La funzione di input per i neuroni esterni è la somma pesata dei suoi input
    - la funzione di input per i neuroni nascosti è una funzione di distanza tra il vettore di input e quello dei pesi
    - funzione di distanza: Minkowski family (manhattan o city block, euclidea, ecc…)
- La funzione di attivazione per i neuroni esterni è una funzione lineare
    - la funzione di attivazione per i neuroni nascosti è una funzione radiale, monotona decrescente (rettangolo, triangolo, coseno, gaussiana, ecc…)
    - la grandezza della regione è data da un raggio di riferimento σ
- Apprendimento supervisionato:
    - dato un task con dei pattern di apprendimento e un neurone nascosto per ciascun pattern (semplice)
        - se la funzione di attivazione è una gaussiana, il raggio di riferimento è scelto euristicamente
    - inizializzazione delle connessioni creando la matrice A*vettore pesi = vettore output
        - risolvibile invertendo A (matrice dei valori di output) per ricavare il vettore dei pesi
        - se l'inizializzazione porta già a una soluzione ottimale, il successivo allenamento non è necessario
    - radial basis function generale:
        - seleziona un subset dei pattern di apprendimento come centri
            - tutti i dati come centri
                - soltanto il raggio e i pesi di output devono essere determinati (ma potrebbe risultare complesso)
                - i valori di output possono essere calcolati perfettamente
            - random subset come centri

        - veloce e soltanto il raggio e i pesi di output devono essere determinati (le performance dipendono però dai valori scelti)
    - clustering
        - c-means clustering
            - scelta di un numero c scelto dall'utente (numero di cluster)
            - inizializza il centro dei cluster casualmente
            - assegna i data point al più vicino cluster
            - calcola i nuovi cluster dati dal vettore medio dei data point assegnati
            - ripetere il processo fintanto che i nuovi cluster non sono uguali ai vecchi
    - il vettore dei pesi per i neuroni nascosti sono i pattern
        - quello per i neuroni esterni è dato invertendo la matrice (?)

Learning Vector Quantization

- E' una rete neurale feed-forward a 2 strati (o una RBF con hidden layer=output layer)
- La funzione di input di ogni neurone di output è una funzione di distanza del vettore di input e il vettore dei pesi
- La funzione di attivazione di ogni neurone di output è una funzione radiale
- La funzione di output dello strato di output considera l'attivazione di tutti i neuroni di output controllando il massimo (potrebbe essere scelto casualmente un neurone a 1 e gli altri a 0 se ci sono più massimi)
- Regole di aggiornamento (dei punti e dei vettori o cluster di riferimento) per LVQ:
    - attrazione: si avvicina il punto al vettore
    - repulsione: si allontana il punto dal vettore
    - avendo un tasso di apprendimento fissato, si può giungere a delle oscillazioni
        - si utilizza un time-dependent learning rate
    - si aggiorna il vettore vicino al dato ma anche i due più vicini
        - può portare i vettori ad allontanarsi sempre di più
    - nella Soft LVQ, si utilizza una gaussiana ???

Self Organizing Maps

- Una SOM (o Kohonen feature map) è una rete neurale feed-forward a 2 strati simile alla LVQ con connessioni solo tra neuroni vicini
- La funzione di input di ogni neurone di output è una funzione di distanza del vettore di input e il vettore dei pesi
- La funzione di attivazione di ogni neurone di output è una radial function
    - funzione monotona decrescente
- La funzione di output di ogni neurone di output è la funzione identità

    - l'output è discretizzato secondo il principio winner takes all, cioè i neuroni competono per l'attivazione e solo quello con l'input più forte si attiva (in contrapposizione con error backpropagation)
- Per ogni neurone di output è definita una relazione con i vicini
    - formano una griglia (quadrata o esagonale)
    - nel caso l'input non sia 2d, si cerca di preservare la topologia (ciò che è vicino rimane vicino) ma non le distanze effettive (es. globo in 2d)
    - la topologia si trova rispettando la vicinanza, data da una radial function come funzione di vicinanza, un time dependent learning rate e un time dependent neighborhood radius
    - la grandezza del vicinato si riduce nel tempo
- Apprendimento:
    - inizializza il vettore dei pesi dei neuroni
        - ponendo il vettore iniziale nell'input e scegliendo casualmente degli esempi di apprendimento
    - seleziona un training sample
    - trova il neurone vincente (il neurone con il vettore di riferimento più vicino)
    - computa il raggio e il learning rate e adatta i vicini rispetto al neurone vincente
    - può fallire se il learning rate o il raggio sono troppo piccoli

Hopfield networks

- Caratteristiche:
    - tutti i neuroni sono sia input che output
    - non ci sono neuroni nascosti
    - ogni neurone riceve input da tutti gli altri neuroni
    - un neurone non è collegato con se stesso
    - i pesi delle connessioni sono simmetrici
- La funzione di input di ogni neurone è la somma pesata degli output di tutti gli altri neuroni
- La funzione di attivazione di ogni neurone è una funzione di soglia (1 oppure -1)
- La funzione di output di ogni neurone è la funzione identità
- La matrice generale dei pesi ha 0 sulla diagonale maggiore, altrimenti il peso tra i due neuroni
- L'aggiornamento
    - converge se i neuroni sono aggiornati sequenzialmente (può cambiare lo stato raggiunto a seconda dell'ordine) in al più $2*2^n$ passi se aggiornati ciclicamente
        - se dopo aver attraversato n neuroni nessuna attivazione cambia, la computazione è conclusa e si è raggiunto uno stato stabile
        - altrimenti l'energia diminuisce e quindi un nuovo stato avrà energia minore del precedente
    - può divergere se aggiornati in parallelo
- Memoria associativa
    - salvando i pesi utilizzati per raggiungere uno stato stabile, si creano dei pattern

- Possibili ottimizzazioni:
    - la funzione di attivazione dei neuroni è un numero intero (può rimanere bloccato su massimi locali ma se si definisce più probabile passare da massimo a minimo si evita il problema)
    - annichilimento tramite temperatura sempre più bassa per raggiungere uno stato stabile

Boltzmann machines

- Simili alle reti di Hopfield
    - cambia come vengono aggiornati i neuroni ed è possibile definire una funzione che assegna energia a un neurone
    - può avere neuroni nascosti
- Una distribuzione di probabilità sugli stati è definita sulla distribuzione di Boltzmann
- La probabilità di attivazione di uno stato è una funzione logistica della differenza di energia tra il suo stato attivo e inattivo
- Abbassando lentamente la temperatura del sistema viene simulato l'annichilimento è quindi il raggiungimento di uno stato stabile
- Addestramento:
    - si può calcolare la distribuzione di probabilità in modo efficiente se gli esempi forniti sono di una distribuzione di Boltzmann
    - si sceglie una misura per la differenza tra due distribuzioni di probabilità e si utilizza la discesa del gradiente per minimizzare l'errore
    - ogni step ha 2 fasi:
        - positiva: i neuroni visibili sono aggiornati casualmente a un data point e i neuroni nascosti sono aggiornati fino a raggiungere l'equilibrio termico
        - negativa: tutte le unità sono aggiornate fino al raggiungimento dell'equilibrio termico
- Regole di aggiornamento:
    - se il neurone è più attivo quando i dati sono presentati rispetto alle esecuzioni, il valore di soglia va abbassato
    - se i neuroni sono più attivi quando i dati sono presentati rispetto alle esecuzioni, il peso delle loro connessioni va aumentato
- La rete è inefficiente per grandi input
    - macchina di Boltzmann ridotta, con grafo bipartito e non completamente connesso
    - le connessioni tra neuroni sono possibili solo tra neuroni di gruppi differenti
    - i neuroni di input sono neuroni di output e viceversa ma i neuroni nascosti cambiano per ogni neurone

Definizioni

- Un insieme di punti in uno spazio euclideo è detto **convesso** se è un insieme non vuoto e connesso (cioè è una regione) e se per ogni coppia di punti, l'insieme dei punti che costituisce il segmento che collega questi punti è anch'esso nell'insieme

- L’**inviluppo convesso** (convex hull o involucro convesso) di un insieme di punti X in uno spazio euclideo è il più piccolo insieme convesso che contiene X
    - è l’intersezione di tutti gli insiemi convessi che contengono X
- Due insiemi di punti in uno spazio euclideo sono detti **linearmente separabili**, se e solo se esiste almeno un punto, linea, piano o iperpiano (a seconda delle dimensioni dello spazio euclideo), tale che tutti i punti di un insieme sono da un lato mentre tutti i punti dell’altro insieme sono dall’altro lato di questo punto, linea, piano o iperpiano. Se esiste, gli insiemi dei punti possono essere separati da una funzione lineare di decisione
    - se e solo se i loro inviluppi convessi sono disgiunti (cioè non hanno punti in comune)

Fuzzy systems

Introduzione

- Un fuzzy set di un insieme X è un mapping da X in [0, 1] che assegna ad ogni elemento di X un grado di appartenenza all'insieme fuzzy stesso
    - 1 = appartenenza totale a un predicato M
    - 0 = assolutamente nessuna appartenenza a un predicato M
- I gradi di appartenenza sono definiti per convenzione e sono dipendenti dal contesto
- Utilizzi:
    - classificazione e analisi di dati
    - problemi di decisione
    - ragionamento approssimato
- Semantica del grado di appartenenza
    - similarità
        - $\mu(u)$ è il grado di prossimità di u dagli elementi di μ
        - usato per classificazione di pattern, cluster analysis e regressione
    - preferenza
        - μ rappresenta sia gli oggetti più o meno preferiti che i valori di una variabile decisionale X
        - $\mu(u)$ rappresenta sia l'intensità di preferenza in favore di un oggetto u che la fattibilità di prendere u come valore di X
        - usato nel design ingegneristico e nei problemi di schedulazione
    - possibilità
        - $\mu(u)$ è il grado di possibilità che il parametro μ abbia valore u
        - usato nelle intelligenze artificiali
- In un fuzzy set basato su valori di verità a n valori, ogni norma triangolare (cioè una funzione commutativa, associativa e con un elemento identità) può essere usata per rappresentare la congiunzione e ogni conorma triangolare può essere usata per rappresentare la disgiunzione
- Una relazione fuzzy generalizza a vari gradi di forza l'associazione o l'interazione tra elementi sui gradi di appartenenza (facendo della relazione booleana un caso particolare)
    - è un fuzzy set definito sulle tuple $(x_1, \ldots, x_n)$ che possono avere vari gradi di appartenenza nella relazione
    - il prodotto cartesiano di fuzzy set è una relazione fuzzy definita dalla funzione di appartenenza basata sul minimo (per ogni valore si controlla il suo valore in ogni insieme fuzzy)
        - $\mu_{AxB}(x, y) = T[\mu_A(x), \mu_B(y)] \ \forall x \in X \ \forall y \in Y$
- Una relazione è detta di equivalenza se è:
    - riflessiva: se e solo se $\forall x \in X : R(x, x) = 1$
    - simmetrica: se e solo se $\forall x, y \in X : R(x, y) = R(y, x)$
    - transitiva: se soddisfa $\max_{y \in Y} \min \{R(x, y), R(y, z)\}, \ \forall (x, z) \in X^2$
- Un fuzzy controller consiste in:

    - fuzzification interface: riceve i valori in input ed eventualmente li mappa in un dominio più adatto (linguistico oppure fuzzy set)
    - base di conoscenza (sui dati e sulle regole): informazioni sui limiti, regole di trasformazione, fuzzy set con i rispettivi termini linguistici e le regole di controllo del tipo if-then
    - logica decisionale: computa l'output in base all'input e alla conoscenza di base
    - defuzzification interface: determina i valori di output convertendo i valori fuzzy nel dominio appropriato (es. booleano)
        - max criterion: scegliere un valore arbitrario $y \in Y$ tale che si raggiunga il massimo valore di appartenenza (applicabile per insiemi arbitrari anche diversi da R ma non deterministico e porta ad azioni discontinue)
        - mean of maxima: dato Y come intervallo, Y_{MAX} è l'insieme delle y tale che l'output è massimo. Il valore di output è la media di Y_{MAX} (può portare ad azioni discontinue)
        - center of gravity: dato Y come intervallo, si trova il centro dell'area data dall'output (di solito ha un comportamento regolare ma una computazione lunga e possibili risultati controintuitivi)
- Tipologie di controller:
    - mamdani control
    - takagi-sugeno control
        - modifica ed estende mandani. Suddivide l'input in partizioni fuzzy ma non l'output
        - le regole del controller sono nella forma R_r: if E_1 is $A_{i1,r}$ and … and E_n is $A_{in,r}$ then $\eta_r = f_r(E_1, \ldots, E_n)$ con f_r: X_1 x … x $X_n \rightarrow Y$ lineare
        - viene computato il valore di verità α_r date r regole e un input $(x_1, \ldots, x_n)$
        - il valore di output calcolato è già crisp e quindi non è necessaria la defuzzification
    - fuzzy control as similarity-based reasoning
        - usa il concetto di relazione fuzzy di equivalenza (relazione di somiglianza)
        - si comporta come il mamdani controller

Fuzzy Clustering

- Apprendimento non supervisionato
    - Si divide il dataset per:
        - gli oggetti simili appartengono allo stesso cluster
        - gli oggetti che appartengono a diversi cluster sono il più possibili dissimili
    - La somiglianza è calcolata tramite una funzione di distanza
        - più è piccola la distanza, più sono simili
    - Costruzione dei cluster (hard c-means)
        - scelta del numero di cluster c
        - inizializzazione con punti random dei c cluster

        - assegnamento degli altri punti al cluster più vicini
        - aggiornamento dei punti del cluster (tramite calcolo del centro di gravità)
        - ripetere il processo fintanto che non cambia il punto del cluster
    - le partizioni dei cluster sono ottimali quando la somma delle distanze quadrate tra i centri e gli elementi è minima
- Nei fuzzy clustering è possibile assegnare un'appartenenza parziale dei punti a un cluster
    - consente di assegnare i punti a più cluster
    - possono essere probabilistici (un punto è presente in ogni cluster con una probabilità > 0 e la somma totale dà 1) e possibilistici (un punto è presente in ogni cluster con una probabilità > 0 ma la somma non deve dare necessariamente 1)
        - nei PCM (possibilistic) i punti con bassa appartenenza sono considerati outliers
    - la fuzziness è un parametro m (se = 1 rimane hard clustering, se > 1 è man mano più soft)
- Ottimizzazioni di funzioni:
    - definire condizioni per massimi e minimi locali e risolvere le derivate parziali
    - lagrange multipliers ponendo le derivate delle funzioni di Lagrange a 0
- Fuzzy c-means (FMC)
    - alternativamente ottimizzato
    - ottimizzare U per i parametri del cluster fissati $U_T = J_U(C_T\text{-}1)$
    - ottimizzare C per il grado di appartenenza fissato $C_T = J_C(U_T)$
    - l'aggiornamento è dato ponendo la derivata J_f a 0 (l'equazione risultante è dunque indipendente dalla misura della distanza)
- Problemi:
    - numero di cluster
        - scegliere un numero c e vedere se è ottimale controllando se i dati sono separati il più possibile, se c'è un numero minimo di cluster e se i punti di ogni cluster sono vicini tra loro
    - forma e locazione dei cluster non noti a priori
    - differenti dati richiedono differenti caratteristiche
- Algoritmi di distanza
    - gli algoritmi euclidei consentono di creare soltanto cluster sferici
    - fuzzy gustafson-kessel
        - distanza euclidea rimpiazzata con la distanza di Mahalanobis
        - si possono imporre vincoli basati sulla forma dei cluster oppure per usare solo matrici diagonali
        - preferito quando il clustering è applicato per la generazione di regole fuzzy
        - si può mettere un vincolo sul numero dei cluster imponendo una condizione sul determinante della matrice
    - fuzzy shell clustering
        - estrae forme geometriche diverse per i cluster
    - kernel-based

        - modifica la funzione di distanza per riconoscere dati non vettoriali come sequenze, alberi e grafi
        - basati su una funzione di mapping da uno spazio di input a uno spazio di Hilbert
        - kernel method = algoritmi con prodotti scalari tra dati
        - bisogna scegliere un kernel adatto e i suoi parametri e i centri dei cluster appartengono allo spazio di Hilbert ma non è noto esplicitamente
- Noise clustering:
    - ai cluster si aggiunge un noise cluster che si occupa di raggruppare tutti i noisy data e gli outlier
    - il suo centro è equidistante da tutti i punti (che quindi hanno la stessa probabilità di appartenere a questo cluster)

Neuro fuzzy systems

- A differenza delle reti neurali, i fuzzy system
    - hanno a che fare col ragionamento ad alto livello e non a strutture a basso livello
    - usano informazioni linguistiche relative al dominio e non si basano sui dati
    - non imparano e non si adattano a nuovi ambienti
    - sono basati sul linguaggio naturale e non sono black box
- I neuro fuzzy system combinano la computazione parallela e la capacità di imparare delle reti neurali con la rappresentazione della conoscenza ad alto livello e le abilità di spiegazione dei fuzzy system
- I due sistemi lavorano indipendentemente (rete neurale e controller)
    - la rete neurale genera certi parametri (offline) oppure
    - la rete neurale ottimizza certi parametri (online)
- I due sistemi sono combinati (modello ibrido)
    - gli insiemi fuzzy e regole fuzzy sono mappati in una rete neurale
    - la struttura è integrata, non c'è bisogno di comunicazione e sia apprendimento offline che online sono possibili
- Gli insiemi fuzzy antecedenti alle regole fuzzy possono essere
    - pesi delle connessioni
        - i neuroni del primo strato rappresentano la regola
    - funzioni di attivazione
        - i neuroni del primo strato rappresentano l'insieme mentre i neuroni del secondo strato la regola
- Algoritmo:
    - ogni variabile di input è un neurone nello strato di input
    - ogni variabile di output è un neurone nello strato di input
    - per ogni fuzzy set, crea un neurone nel primo strato nascosto collegato allo strato di input
    - per ogni regola, crea un neurone nel secondo strato nascosto, specifica una t-norm per applicare la regola di attivazione
    - collega ogni regola con l'insieme fuzzy corrispondente

    - per mamdani: per ogni neurone di regola, collegalo al neurone di output corrispondente e come peso utilizza il fuzzy set conseguente la regola
    - per takagi: per ogni neurone di regola, crea un altro neurone che computa l'output della corrispondente regola
    - la rete può essere addestrata tramite error backpropagation

Definizioni

- Una notazione è detta **imprecisa** se il suo significato non è fissato
    - es. fino a un certo livello
    - es. gradualità
- Una proposizione è detta **imprecisa** se contiene predicati graduali
    - es. vero fino a un certo livello
    - es. quasi
- L'imprecisione non è l'incertezza (data da probabilità o possibilità)
- Un **insieme** è ogni collezione di definiti e distinti oggetti
    - $x \neq \{x\}$
    - l'insieme di tutti i possibili sottoinsiemi è 2^X
    - $\emptyset$ è l'insieme $\emptyset$
- L'insieme di tutti i possibili insieme fuzzy di un insieme X è F(X)
    - $F(X) = \{\mu \mid \mu: X \rightarrow [0, 1]\}$
- La rappresentazione orizzontale di un fuzzy set è detta **α-cuts**
    - $[\mu]_\alpha = \{x \in X \mid \mu(x) \geq \alpha\}$ = α-cut di μ
    - $[\mu]_\alpha = \{x \in X \mid \mu(x) > \alpha\}$ = strict α-cut di μ
- Il **support** S(μ) di un fuzzy set $\mu \in F(X)$ è il crisp (boolean) set che contiene tutti gli elementi di X che che hanno un grado di appartenenza diverso da 0
    - $S(\mu) = [\mu]_{\neg 0} = \{x \in X \mid \mu(x) > 0\}$
- Il **core** C(μ) di un fuzzy set $\mu \in F(X)$ è il crisp set che contiene tutti gli elementi di X che hanno un grado di appartenenza a 1
    - $C(\mu) = [\mu]_1 = \{x \in X \mid \mu(x) = 1\}$
- L'**altezza** di un fuzzy set $\mu \in F(X)$ è il alto grado di appartenenza ottenuto da qualsiasi elemento di quell'insieme
    - $h(\mu) = \sup_{x \in X}\{\mu(x)\}$
    - un fuzzy set è detto normale se e solo se h(μ) = 1, altrimenti è detto subnormale
- Dato X come spazio vettoriale, un fuzzy set $\mu \in F(X)$ è detto **fuzzy convex** se i suoi α-cuts sono convessi per ogni $\alpha \in (a, 1]$
    - la funzione di appartenenza di un convex fuzzy non è un funzione convessa (ma concava)
- μ è un **fuzzy number** se e solo se μ è normale e $[\mu]_\alpha$ è limitato, chiuso e convesso $\forall$ $\alpha \in (0, 1]$
- Un insieme di primitive logiche è **completo** se ogni funzione logica può essere composta da un numero finito di primitive $\{\neg, \wedge, \vee, \rightarrow\}$
- Una formula logica è una **tautologia** se è vera per ogni valore di verità (è una **contraddizione** se è sempre falsa)
    - $(A \wedge (A \rightarrow B)) \rightarrow B$ (modus ponens)

    - $(\neg B \land (A \rightarrow B)) \rightarrow \neg A$ (modus tollens)
    - $((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C)$ (sillogismo)
    - usate come regole di inferenza (per inferire il lato destro dell'implicazione come vero)
- Logiche a **n valori** (L_n)
    - $\neg a = 1 - a$
    - $a \land b = \min(a, b)$
    - $a \lor b = \max(a, b)$
    - $a \rightarrow b = \min(1, 1 + b - a)$
    - $a \leftrightarrow b = 1 - |a - b|$

Evolutionary algorithms

- Un problema di ottimizzazione ($\mathbf{\Omega}$, f, $>$) dato uno spazio di ricerca $\mathbf{\Omega}$, una funzione di valutazione f: $\mathbf{\Omega} \rightarrow$ R che assegna una valutazione a tutti i possibili candidati, una di comparazione $> \in \{<, >\}$
    - l'insieme delle soluzioni ottimali è H $\subseteq \mathbf{\Omega}$ è definito come H = {x $\in \mathbf{\Omega}$ | $\forall$ x' $\in \mathbf{\Omega}$: f(x) ≥ f(x')}
- Calcolo soluzioni:
    - soluzione analitica: efficiente ma raramente applicabile
    - esplorazione esaustiva: inefficiente e applicabile quindi solo per insiemi piccoli
    - ricerca casuale: inefficiente ma sempre applicabile
    - ricerca guidata: elementi simili in $\mathbf{\Omega}$ hanno valori di funzione simili (precondizione)
- Elementi necessari per un algoritmo evolutivo:
    - un encoding per i possibili candidati
    - un metodo per creare la popolazione iniziale
    - una funzione di valutazione (fitness function) (rappresenta l'ambiente e specifica le qualità dell'individuo. Spesso è la funzione di ottimizzazione e può contenere elementi aggiuntivi)
    - un metodo di selezione relativo alla fitness function (seleziona le caratteristiche che saranno trasmesse o meno alla generazione successiva)
    - un insieme di operatori genetici per modificare i cromosomi (tramite mutazione, cioè cambi casuali di alcuni geni, oppure tramite crossover, cioè ricombinazione di cromosomi)
    - altri parametri (grandezza della popolazione, probabilità di mutazione, ecc…)
    - criterio di terminazione (numero di generazioni, …)
- Una funzione di decoding dec: G $\rightarrow \mathbf{\Omega}$ è una trasformazione di genotipo G in un fenotipo $\mathbf{\Omega}$
- Un individuo A è una tupla (A.G, A.S, A.F) contenente la soluzione candidata (genotipo A.G), le informazioni aggiuntive opzionali A.S e una qualità della valutazione A.F = f(dec(A.G))
    - un genotipo A.G $\in$ G di un individuo A
    - informazioni aggiuntive o parametri di strategie A.S $\in$ Z (spazio di tutte le possibili informazioni aggiuntive come i parametri relativi agli operatori genetici)
    - qualità o fitness A.F $\in$ R
- Operatori genetici:
    - modifica del genoma
    - un operatore di mutazione, applicato su un G-encoded optimization problem e Z), è definita dalla funzione di mapping mut^E: G x Z $\rightarrow$ G x Z
    - un operatore di ricombinazione con r ≥ 2 genitori e s ≥ 1 discendenze (r, s $\in$ N) è definita dalla funzione di mapping rek^E: $(G \times Z)^r \rightarrow (G \times Z)^s$
- Operatori di selezione:

    - input: popolazione di r individui di cui s sono stati scelti
    - la selezione non cambia o crea alcun individuo
    - la selezione definisce gli indici basati unicamente dal fitness
- Un algoritmo evolutivo su un problema di ottimizzazione (E, f, $\succ$) è una 8-tupla (G, dec, mut, rek, IS_{parent}, $IS_{environment}$, μ, λ)
    - μ: numero di individui dei parenti
    - λ: discendenze per generazioni

Meta heuristic

- La meta euristica è un metodo algoritmico per risolvere approssimativamente un problema di ottimizzazione combinatoria
    - definisce una sequenza di step astratti da implementare per ogni problema
- Il local search method è un caso speciale di algoritmo evolutivo
    - popolazione = 1 con varie conseguenze
    - c'è l'operatore di mutazione ma non quello di ricombinazione (solo un individuo, ne servono almeno 2)
    - utilizzando un gradiente ascendente o discendente, cioè un'operazione differenziale che crea un vettore, questo può essere usato per identificare un punto di massimo facendo un piccolo step nella direzione del gradiente
        - la distanza dello step è ottenuta in modo adattiva nel tempo altrimenti o la computazione è troppo lunga o oscilla
        - si può rimanere bloccati in un massimo locale, però si può provare con esecuzioni multiple per risolvere il problema
    - se f non è differenziabile, si prende un punto casuale e si controllano i vicini
        - si può rimanere bloccati in un massimo locale
    - se si applica un annichilimento simulato, cioè muoversi dal basso verso l'alto è più probabile del contrario
        - un risultato migliore va sempre bene
        - uno peggiore può essere accettato a seconda della differenza di qualità e della temperatura
            - si può anche applicare un valore di soglia per accettarlo o meno
            - oppure un lower bound senza alcun confronto con il valore precedente
            - oppure un lower bound crescente (cambia dopo ogni miglioramento)
- Il tabu search è un local search method con una coda (FIFO) per evitare di ricreare degli individui
    - ogni entry è una soluzione completa
    - le mutazioni non sono permesse
    - si può rompere con delle liste fifo con proprietà migliori o da un nuova miglior qualità ???
- Gli algoritmi memetici combinano gli algoritmi population based (lenti ma che considerano l'intero spazio) e il local search (veloci ma suscettibili ai local optima)

    - i memes sono gli elementi del comportamento che possono essere acquisiti individualmente (in contrasto ai geni)
    - ottimizzazione accelerata fortemente
    - la mutazione si blocca spesso nei local optima
    - la ricombinazione ha una situazione iniziale fissata
    - non tutto lo spazio potrebbe essere raggiungibile
- L'evoluzione differenziale non adatta la grandezza dello step in base a delle strategie ma considera la relazione degli individui nella popolazione
    - de-operator: combinazione di ricombinazioni e mutazione
    - un nuovo individuo rimpiazza il genitore solo se ha un fitness migliore
- Una scatter search prevede
    - una popolazione con soluzioni candidate, operatori di variazione, una pressione di selezione, local search
    - un metodo deterministico, un'ampia esplorazione dello spazio, una grande inizializzazione, una sistematica generazione di nuovi individui
    - iterazione:
        - generazione di nuovi individui e selezione di quelli che garantiscono una varietà maggiore
        - ricombinazione di tutte le coppie di individui scelti, scelta dei migliori e ripete il processo
    - parametri raccomandati: grandezza della popolazione, numero dei migliori α, espansione dei migliori β
- Un algoritmo culturale aggiunge informazioni allo strato della cultura che saranno modificate dai migliori individui
    - all'inizio si considera un piccolo sottoinsieme su cui basarsi
    - successivamente si prende il minimo e il massimo del migliore 20% e si accettano solo miglioramenti (eccezione: migliore qualità ma valori peggiori)
    - dimensione spaziale stabile se si considerano solo gli ultimi migliori
    - la mutazione se stabile si orienta verso il miglior individuo, altrimenti adatta lo step

Elementi degli algoritmi evolutivi

Encoding

- L'encoding di una soluzione dipende dal problema considerato
    - non esiste un metodo generale ma solo linee guida
    - la rappresentazione di fenotipi simili dà genotipi simili
        - la mutazione di certi geni risulta in genotipi simili
        - grandi cambiamenti del genotipo produce un simile o migliore fenotipo
    - fitness simile da candidati simili
        - se c'è un encoding fortemente epistatico, non ci sono regolarità e una mutazione può portare a casuali cambiamenti di fitness. Il problema di ottimizzazione non è facilmente risolvibile da un EA
        - se c'è un encoding scarsamente epistatico, ci sono metodi migliori di risoluzione

        - l'epistasi è una proprietà dell'encoding, non del problema
    - chiusura su Ω con gli operatori evoluzionari
        - lo spazio di ricerca non è chiuso se un nuovo cromosoma non può essere interpretato o decodificato
            - si possono usare meccanismi di riparazione per ovviare al problema che però introducono un nuovo termine di penalità che riduce il loro valore di fitness
        - oppure una soluzione non rispetta determinati vincoli di base
        - oppure una soluzione è valutata erroneamente dalla funzione di fitness

Fitness

- Gli individui migliori dovrebbero avere migliori chance di avere una discendenza
- La pressione della selezione è la forza di preferenza degli individui migliori
    - con l'esplorazione dello spazio è bassa ma migliori chance di trovare un optima globale
    - con l'analisi degli individui migliori è alta e converge a un migliore
- Le metriche considerate per la pressione sono:
    - il numero di generazioni fino a convergere (bassa pressione all'inizio, alta pressione dalla fine)
    - la differenza media tra la qualità prima e dopo la selezione (intensità di selezione)
- Metodi di selezione:
    - roulette-wheel
        - metodo più conosciuto e proporzionato
        - calcola il fitness relativo che viene visto come la probabilità di essere selezionato
        - la computazione di ogni somma è costosa ed è difficile da parallelizzare
        - gli individui con un alto fitness potrebbero dominare la selezione (scompare la diversità della popolazione)
        - effettua una ottimizzazione locale (preferibile nelle generazioni successive e non all'inizio)
- La funzione di fitness si può adattare tramite:
    - linear dynamical scaling che utilizza il minimo delle ultime generazioni e non solo l'ultimo
    - σ-scaling che prevede l'uso di due parametri aggiuntivi
    - dipendenza temporale (fattore temporale usato come esponente)
    - boltzmann selection (la temperatura dipende dal tempo)

Selezione

- La selezione degli individui è proporzionale al fitness ma è casuale
    - non c'è garanzia che gli individui migliori siano considerati per la prossima generazione
    - si discretizza il range di fitness e si scelgono i candidati migliori della media

    - risolve il problema della varianza
- La valutazione del voto considera le generazioni passate e anche gli individui migliori hanno al più un numero k di generazioni
    - viene usato come il metodo della roulette
- Rank-based selection
    - si ordinano gli individui in ordine decrescente in base al fitness
    - un rank è assegnato ad ogni individuo e si definisce una probabilità rispetto ad esso
    - roulette-wheel basata sulla distribuzione di probabilità
    - evita il problema della dominanza e regola la pressione della selezione con la distribuzione di probabilità
    - però è necessario ordinare gli individui (complessità n logn con n = |P|)
- Tournament selection
    - estrai k individui casualmente nella popolazione
    - gli individui gestiscono il torneo e chi vince avrà una generazione
    - tutti i partecipanti compreso il vincitori ritornano nella popolazione
    - evita il problema della dominanza e regola la pressione della selezione con la grandezza del torneo
- Elitism
    - i migliori individui entrano nella nuova generazione
    - non c'è protezione dalle modifiche degli operatori genetici (data dalla normale selezione)
    - il fitness degli individui migliori può decrescere da una generazione all'altra
    - la generazione data da combinazione o mutazione può rimpiazzare i genitori
        - se il fitness è migliore
    - si ottiene una convergenza migliore al local optimum
    - si può bloccare in un local optima se non è possibile una degradazione locale
- Crowding
    - le generazioni successive dovrebbero rimpiazzare gli individui più simili a loro (richiede una funzione di somiglianza o distanza)
    - con la combinazione si assegna ciascun figlio al genitore più simile e si prende il migliore tra i due
    - si evitano così computazioni tra simili in quanto si considera solo una parte della popolazione
- Caratteristiche dei modelli di selezione:
    - statico: probabilità che la selezione rimanga costante
      dinamico: probabilità che la selezione cambi
    - estinzione: probabilità che la selezione sia 0
      preservativo: tutte le probabilità di selezione devono essere > 0
    - pure-bred: individui possono avere discendenza solo in una generazione
      under-bred: individui possono avere discendenza in più generazioni
    - right: tutti gli individui si possono riprodurre
      left: gli individui migliori potrebbero non riprodursi
    - generazionale: i genitori sono fissati fintanto che tutte le discendenze non sono create
      al momento: le generazioni create rimpiazzano i loro genitori

Operatori genetici

- Sono applicati a una frazione di individui scelti
- Genera mutazioni e ricombinazioni di soluzioni candidate già esistenti
- Classificazione:
    - genitore singolo: mutazione
        - cambia il fitness (funzione) della soluzione candidata il meno possibile
        - esplorazione: casuale oppure nelle regioni più lontane dello spazio
        - sfruttamento: miglioramento locale della soluzione e inclusione del vicinato fenotipico
        - standard mutation: scambio di forma/valore di un gene di un altro allele
        - pair swap: scambio delle forme/valori di due geni in un cromosoma
    - genitore doppio: combinazione (crossover)
        - one-point crossover: determinazione di una casuale linea di taglio e scambio dei geni dal punto in poi
        - two-point crossover: determinazione di due casuali linee di taglio e scambio dei geni tra i due punti
        - n-point crossover: generalizzazione dei metodi precedenti alternando scambiare e tenere tra i punti
        - uniform crossover: per ogni gene determina se tenere o scambiare data una probabilità di scambio
        - shuffle crossover: permutazione dei geni e one-point crossover e unmixing
        - uniform order-based crossover: simile a uniform crossover ma per ogni gene determina se tenere o lasciare. Gli spazi vuoti sono riempiti dall'ordine di presenza nell'altro cromosoma
        - edge recombination: il cromosoma è rappresentato come un grafo (catena o anello) e ogni gene contiene vertici con i vicini nel cromosoma. I vertici dei grafi dei due cromosomi sono mischiati preservando le informazioni di vicinanza
    - genitori multipli
        - diagonal crossover: simile a 1-2-n-point crossover ma valido per più genitori. k genitori: k-1 crossover points diagonalmente sui punti di intersezione. Porta a una forte esplorazione dello spazio con un gran numero di genitori (10/15)
- Gli operatori non dovrebbero creare soluzioni scorrette
- Caratteristiche degli operatori di crossover:
    - positional bias: la probabilità che due geni siano ereditati dallo stesso genitore dipende dalla posizione relative dei geni nel cromosoma indesiderato poiché può rendere la disposizione dei geni cruciale per il successo e il fallimento dell'algoritmo
    - distributional bias: se la probabilità che un certo numero di geni è scambiato tra i cromosomi dei genitori non è lo stesso per tutti i numeri di geni

indesiderato poiché causa soluzioni parziali di differente lunghezza ad avere differenti chance di progredire alla prossima generazione
- Interpolating recombination
    - miscela i tratti dei genitori in modo tale da avere una generazione con nuovi tratti (non si esplora tutto lo spazio ma si concentra in un'area principale)
    - per esplorare lo spazio all'inizio si usa una forte e casuale mutazione che preserva la diversità
- Extrapolating recombination
    - cerca di estrapolare informazioni da molteplici individui e crea una prognosi nella direzione che si aspetta possa produrre miglioramenti
    - crea nuova alleli e potrebbe cambiare lo spazio di ricerca $\mathbf{\Omega}$
    - l'influenza della diversità è difficilmente comprensibile

Adaptation strategies

- Un miglioramento del fitness di un individuo A $\in$ G rispetto a un altro individuo B $\in$ G è definito come |B.F - A.F| se B.F > A.F, altrimenti 0
- Il miglioramento relativo atteso di un operatore mut relativo all'individuo A è definito come E(improvement(A, mut(A))
- Gli operatori di mutazione locali sono
    - inversione di una sottosequenza
    - scambio ciclico di 3 geni
- Dove avviene una mutazione è importante in quanto
    - se è molto locale sono coperti soltanto i valori di fitness nella vicinanza (inverting mutation)
    - se è meno locale è coperto un più ampio raggio di valori (triple exchange)
- La qualità di una mutazione non può essere giudicata indipendentemente dal livello attuale di fitness
    - più ci si avvicina al migliore più bisogna usare operatori locali
- Tipologie
    - predefined adaptation: definire i cambiamenti prima
    - adaptive adaptation: si definisce una metrica per stabilire la frazione di mutazioni che hanno apportato dei miglioramenti
    - self adaptation: si utilizzano informazioni aggiuntive sugli individui

Swarm/population based optimization

- In un'intelligenza a sciame (swarm), gli individui hanno capacità ridotte e si scambiano informazioni per coordinarsi
- Tecniche:
    - genetic/evolutionary algorithms: basato sullo scambio di informazioni dato dalla ricombinazione dei genotipi
    - population based incremental learning: basato sullo scambio di informazioni dato dalla prevalenza nella popolazione
    - particle swarm optimization: basato sullo scambio di informazioni dato dall'aggregazione di singole soluzioni

    - ant colony optimization: basato scambio di informazioni dato dalla manipolazione dell'ambiente

Evolutionary algorithms su stringhe

- Uno schema h è una stringa di lunghezza L sull'alfabeto {0, 1, *} tale che h appartenga a $\{0, 1, *\}^L$ con * come wildcard
- Un cromosoma $c \in \{0, 1\}^L$ è un match per lo schema h se e solo se coincide con h per tutte le posizioni dove h è 0 o 1
- La selezione è influenzata dal numero di cromosomi che matchano lo schema (con la relativa media di fitness) rispetto al numero totale di cromosomi (con la relativa media di fitness)
- La mutazione è influenzata dalla probabilità di preservare il match dopo la mutazione
- La ricombinazione è influenzata dalla probabilità di scegliere un punto con valori fissati da una parte (0 o 1 relativi allo schema) e valori liberi dall'altra

Genetic Programming

- La programmazione genetica è basata su:
    - descrivere una soluzione collegando certi input a certi output
    - cercare un programma di match
    - rappresentare i programmi come alberi di parsing
- La rappresentazione delle soluzioni candidate data da una funzione
- La grammatica, che differisce per ogni problema, per descrivere un linguaggio è definita da:
    - F: insieme di simboli e operatori
    - T: insieme di simboli terminali (costanti e variabili)
- La programmazione genetica può risolvere problemi in modo efficace ed efficiente se gli insiemi F e T sono sufficienti/completi da assicurare che un programma può essere trovato
    - trovare l'insieme minimo è un problema spesso NP-hard
- I cromosomi sono espressioni, cioè composizione di elementi di $F \cup T$ e parentesi
- Algoritmo:
    - generare una popolazione iniziale di espressioni casuali
        - parametri: massima profondità e massimo numero di nodi
        - metodi: crescita, pieno, a metà
    - valutare queste espressioni calcolando il fitness
    - selezione con una strategia degli algoritmi evolutivi
        - non ottimizzano l'organismo ma l'intero processo evolutivo (suscettibili a mutazioni, step di mutazione, velocità di evoluzione, …)
            - lo step può basarsi su una varianza piccola (local search) o alta varianza (global search). La varianza si adatta se ⅕ della discendenza è migliore o meno
        - elite principle: i genitori si possono evolvere e la selezione è sulle discendenze. Solo i migliori tra genitori e discendenti entra nella prossima generazione

                - plus-strategy: si prendono gli x individui migliori tra genitori e discendenti. Ci sono solo miglioramenti ma si può bloccare in un local optima
                - comma strategy: si creano più discendenti e si perdono i genitori. Aumenta la diversità e permette di non bloccarsi su un local optima
        - applicazione di operatori genetici
            - crossover: scambio di due sottoespressioni (sottoalberi)
            - mutation: scambio di una sottoespressione con una generata casualmente
            - clonale reproduction: duplicazione di un individuo
- Bisogna prevedere e prevenire gli introni, cioè sequenze che DNA che non hanno alcuna informazione utile (nel caso delle espressioni, a + (1-1))
    - ricombinazione riproduttiva: generare più figli e scegliere il migliore
    - ricombinazione intelligente: sceglie i punti di crossover in modo selettivo
    - cambiamenti minimi continui: cambiamento della funzione di valutazione può rendere attivi programmi inattivi

Multi criteria optimization

- L'ottimizzazione considera più obiettivi che potrebbero anche essere in conflitto tra loro
    - per ogni criterio è definita una funzione
- Approccio semplice:
    - somma di ogni criterio per un peso ad esso associato
- Aggregazione di preferenze:
    - poiché non esiste una funzione che tenga in considerazione tutte le caratteristiche richieste, si crea un ordine in scala delle preferenze
- Applicazione: pareto-optimal
    - si cambia soltanto se nessuna preferenza peggiora ma una migliora

Parallel Evolutionary Algorithms

- Gli algoritmi evolutivi sono computazionalmente costosi
    - grande popolazione (da poche migliaia a decine di migliaia)
    - grande numero di generazioni (qualche centinaia)
    - vantaggio: soluzione migliore rispetto ad altri approcci
- La popolazione iniziale si può parallelizzare in quanto creata casualmente
    - ma potrebbero crearsi duplicati
- La valutazione di un cromosoma è parallelizzabile in quanto non dipende da altri cromosomi
    - calcolando il fitness o il ranking
    - il risultato deve essere gestito da un'unità centrale
- Selezione:
    - l'elitismo richiede tutta la popolazione e non si può parallelizzare
    - la roulette o rank-based può essere parallelizzata dopo lo step iniziale

    - il torneo si può parallelizzare in quanto indipendenti tra loro
- L’applicazione degli operatori genetici può essere applicata in parallelo in quanto riguarda un numero ristretto di cromosomi (uno o due)
- Se la selezione si parallelizza, si può vedere la popolazione come suddivisa in isole
    - i risultati peggiorano se non si scambiano individui ogni tot tempo, non in ogni generazione
    - la migrazione tra isole è scelta casualmente oppure, se le isole sono organizzate in un grafo, solo tra vicini
    - ad ogni isola si può applicare un algoritmo differente con parametri differenti (avendo però un lower bound per la popolazione)
- Se la selezione si parallelizza, si può vedere la popolazione come suddivisa in una griglia
    - la selezione è limitata ai processori adiacenti (seleziona il migliore dei 4 ed effettua il crossover con se stesso)