1 Abstract

L'idea di base é che un effetto sinergico all'interno dell'ensemble migliora la performance e la tolleranza agli errori.

2 Reti neurali

2.1 Storia

2.2 Struttura

2.3 Funzione d'attivazione

2.4 Loss function

2.5 Back propagation

3 Reti neurali a valori complessi

3.1 Storia

Uno dei primi esempi di rete neurale fu il percettrone di Rosenbatt (1958). Di seguito fu creata ADELINE (Widrow & Hoff, 1960), composta da un singolo layer con un singolo neurone e utilizzava il LMS (least mean square) insieme ad un algoritmo di discesa stocastica del gradiente per la configurazione dei pesi. Ne descriviamo adesso le caratteristiche. Siano $x \in R^N$ gli input e $w \in R^N$ i pesi; dal modello abbiamo:

$$\widehat{y} = x \cdot w. \tag{1}$$

Dato l'output desiderato $y \in R$ possiamo definire la funzione errore e e la $loss\ function\ L$. L'obiettivo é quello di trovare il valore dei pesi che minimizzi L.

$$e = y - \widehat{y} \tag{2}$$

$$L = e^2 \tag{3}$$

$$w \leftarrow arg\,min(L). \tag{4}$$

L'algoritmo LMS ottimizza L usando il gradiente discendente:

$$\nabla L_w := -2ex \tag{5}$$

$$w \leftarrow w + \alpha ex \tag{6}$$

dove α é il tasso di apprendimento.

Widrow, McCool and Ball (1975) hanno esteso l'algoritmo LMS al dominio complesso fornendo la derivazione delle parti reali e immaginarie. Brandwood (1983) generalizzó la teoria applicando il gradiente al numero complesso, senza separarlo in parte reale e parte immaginaria attraverso il gradiente di Wirtinger (1927).

Aggiorniamo quindi il problema nel dominio complesso:

$$L = e\overline{e} \tag{7}$$

$$\nabla L_w := -2e\overline{x} \tag{8}$$

$$w \leftarrow w + \alpha e\overline{x}. \tag{9}$$

Nonostante i risultati di Brandwood, fino a non molto tempo fa la letteratura non applicava il calcolo di Wirtinger, a favore della derivazione separata della parte reale da quella immaginaria.

3.2 Struttura

Esaminiamo un modello $feedforward$ avente un unico hidden layer; abbiamo di conseguenza:

$$h = f\left(W^{(h)}x + b^{(h)}\right) \tag{10}$$

$$\widehat{y} = W^{(o)}x + b^{(o)} \tag{11}$$

con $\theta = \left\{W^{(h)},\ W^{(o)},\ b^{(h)},\ b^{(o)}\right\}$ sono i parametri del modello. Definendo M il numero dei nodi dell'input e N il numero dei nodi dell'output, avremo:

$$\begin{array}{ccc} W^{(h)} \in R^{M\times M} & or & W^{(o)} \in C^{M\times M} \\ b^{(h)} \in R^{M} & or & b^{(o)} \in C^{M} \\ W^{(o)} \in R^{N\times M} & or & W^{(o)} \in C^{N\times M} \\ b^{(o)} \in R^{N} & or & b^{(o)} \in C^{N} \end{array} \tag{12}$$

In tutti i casi gli inut, i pesi e gli output appartengono allo stesso dominio numerico.

3.3 Funzione d'attivazione

Identitá Permette una modellizazione lineare

$$f^{(-)}(x) := x \tag{13}$$

Tangente Iperbolica Eúna funzione sigmoidale e differenziabile. Permette una non linearitá ampiamente utilizzata per l'apprendimento delle reti neurali

$$f^{(\sigma)} := tanh(x) \tag{14}$$

Split reale-immaginario Applica la tangente iperbolica separatamente alla parte reale e alla parte immaginaria:

$$f^{(ri)}(x) := tanh(Re\ x) + i\ tanh(Im\ x) \tag{15}$$

Split ampiezza-fase Applica la tangente iperbolica al modulo del numero complesso, senza modificarne la fase (questa funzione non é differenziabile nel campo complesso)

$$f^{(ap)}(x) := tanh(|x|)\, e^{i\, arg\, x} \tag{16}$$

ModReLU Arjosky nel 2015 propose una variazione alla classica ReLU utilizzata nelle reti neurali reali, definita come segue:

$$ModReLU(z) = ReLU(|z| + b)\, e^{i\theta_z} = \begin{cases} (|z| + b)\, \frac{z}{|z|} & \text{se } |z| + b \geq 0 \\ 0 & \text{altrimenti} \end{cases} \tag{17}$$

dove θ_z é la fase di z e b il bias, apprendibile dalla rete neurale, e necessario per creare una *dead zone* di raggio b attorno all'origine dove il neurone é inattivo. Tale funzione non soddisfa le equazioni di Cauchy-Riemann e quindi non é olomorfa.

CReLU Consiste in una applicazione della funzione ReLu individualmente alla parte reale e alla parte immaginaria:

$$CReLU\,(z) = ReLU\,(Re\,(z)) + i\,ReLU\,(Im\,(z))\,. \tag{18}$$

Questa soddisfa le condizioni di Cauchy-Riemann solo se sia la parte reale che la parte immaginaria sono contemporaneamente strettamente positive o strettamente negative, quindi nell'intervallo $\theta_z \in \left(0, \frac{\pi}{2}\right)$ oppure $\theta_z \in \left(\pi, \frac{3\pi}{2}\right)$.

zReLU Proposta nel 2016 da Guberman e basata anch'essa sulla ReLu, é definita come:

$$zReLU\,(z) = \begin{cases} z & \text{se}\theta_z \in \left[0, \frac{\pi}{2}\right] \\ 0 & \text{altrimenti} \end{cases} \tag{19}$$

3.4 Back propagation

3.4.1 Loss function

La maggior parte della letteratura attuale utilizza come funzione di costo la *mean squarred error*. Dato un target y e l'output ottenuto $\widehat{y}$, entrambi in C^N e l'errore:

$$e := y - \widehat{y} \tag{20}$$

la *complex mean squared loss function* é definita come segue:

$$L(e) = \sum_{i=0}^{N-1} |e_i|^2 \tag{21}$$

$$= \sum_{i=0}^{N-1} e_i \overline{e_i}. \tag{22}$$

La 21 é una funzione a valori reali scalari non negativi, che tende a zero insieme al modulo dell'errore. Savitha, Suresh e Sundararajan proposero di sostituire l'errore 20 con:

$$e := log(\widehat{y}) - log(y). \tag{23}$$

La *loss function* diventerebbe quindi:

$$L(e) = (log\,|\widehat{y_i}| - log\,|y_i|)^2 + (arg(\widehat{y_i}) - arg(y_i))^2 \tag{24}$$

L'equazione 24 ha la proprietá di rappresentare esplicitamente l'ampietta e la fase. Le equazioni 20 e 23 possono essere *error function* appropriate per reti neurali complesse.

3.4.2 Il gradiente complesso ed il calcolo di Wirtinger

Brandwood e Van den Bos formularono le prime derivazioni del gradiente complesso. Wirtinger (1927) forní un formalismo equivalente che rese il calcolo della derivata di funzioni con valori complessi meno oneroso rispetto a funzioni olomorfe e non analitiche , agendo interamente nel campo complesso. Nonostante tale apparente comoditá, solo recentemente si ricominció ad utilizzare il calcolo di Wirtinger per la *backpropagation* di reti neurali complesse.

Definiamo:

$$f(z) := f(z, \overline{z}) \tag{25}$$
$$= g(x, y) \tag{26}$$
$$= u(x, y) + iv(x, y) \tag{27}$$

con $z \in C$, $x, y \in R$ e $z = x + iy$.

Usando la prima definizione avremo le derivate in z e $\overline{z}$ date da:

$$\left.\frac{\partial f}{\partial z}\right|_{\overline{z}\ costante} \tag{28}$$
$$\left.\frac{\partial f}{\partial \overline{z}}\right|_{z\ costante} \tag{29}$$

le quali, espresse in funzioe di x e y, diventano:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right) \tag{30}$$
$$\frac{\partial f}{\partial \overline{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right). \tag{31}$$

Si osservi che la derivata parziale rispetto a $\overline{z}$ é nulla per ogni funzione olomorfa. Richiamiamo le condizioni di esistenza di Cauchy-Riemann per la derivata complessa della funzione $f(z, \overline{z}$ presa in considerazione:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \tag{32}$$
$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}. \tag{33}$$

Se applichiamo le condizioni di Cauchy-Riemann alla derivata di f in $\overline{z}$ 31 notiamo che effettivamente essa si annulla. Le funzioni olomorfe quindi non dipendono esplicitamente da $\overline{z}$. Brandwood dimostró che l'annullarsi di 30 o 31 per una generica $f : C \to R$ é condizione sufficiente e necessaria affinché f abbia un punto stazionario. Per estensione se $f : C^N \to R$ é una funzione a valori reali di un certo vettore $z = [z_0,\ z_1,\ \cdots,\ z_{N-1}]^T \in C^N$ e definiamo il cogradiente e il gradiente coniugato come:

$$\frac{\partial}{\partial z} := \left[\frac{\partial}{\partial z_0}, \frac{\partial}{\partial z_1}, \cdots, \frac{\partial}{\partial z_{N-1}}\right] \tag{34}$$
$$\frac{\partial}{\partial z} := \left[\frac{\partial}{\partial \overline{z_0}}, \frac{\partial}{\partial \overline{z_1}}, \cdots, \frac{\partial}{\partial \overline{z_{N-1}}}\right] \tag{35}$$

allora $\frac{\partial f}{\partial z} = 0$ o $\frac{\partial f}{\partial \overline{z}} = 0$ sono condizioni sufficienti e necessarie per determinare un punto di stazionarietá.

Se f é una funzione di un vettore complesso z, la sua derivata totale é:

$$df = \frac{\partial f}{\partial z} dz + \frac{\partial f}{\partial \overline{z}} d\overline{z}. \tag{36}$$

Se f é reale allora avremmo:

$$df = 2Re\left\{\frac{\partial f}{\partial z} dz\right\}. \tag{37}$$

Definendo ora l'operatore gradiente come:

$$\nabla_z := \left(\frac{\partial}{\partial \overline{z}}\right)^T \tag{38}$$

$$= \left(\frac{\partial}{\partial z}\right)^* \tag{39}$$

puó essere dimostrato, usando la disuguaglianza di Cauchy-Scharz, che f ha il maggior tasso di cambiamento lungo i gradiente. Grazie a queste definizioni possiamo costruire una *cost function* reale con argomenti complessi anche se alcuni elementi della funzione non sono olomorfi. In generale, date le funzioni arbitrarie fe g, combinando gli jacobiani come segue

$$J_{f\circ g} = J_f J_g + J_f^{(c)} \overline{\left(J_g^{(c)}\right)} \tag{40}$$

$$J_{f\circ g}^{(c)} = J_f J_g^{(c)} + J_f^{(c)} \overline{(J_g)}. \tag{41}$$

Supponiamo di avere una funzione composta $(f \circ g \circ h)(z, \overline{z})$, con f la *cost function* reale, g una funzione complessa non olomorfa e h una funzione olomorfa. tenendo a mente che f é una funzione a valori reali di variabile complessa e che h é olomorfa ($\frac{\partial h}{\partial \overline{z}=0}$), applichiamo la *chain rule*:

$$J_f = \frac{\partial f}{\partial g} \tag{42}$$

$$J_f^{(c)} = \frac{\partial f}{\partial \overline{g}} \tag{43}$$

$$J_{f\circ g} = J_f \frac{\partial g}{\partial h} + J_f^{(c)} \overline{\left(\frac{\partial g}{\partial \overline{h}}\right)} \tag{44}$$

$$J_{f\circ g\circ h} = J_{f\circ g} \frac{\partial h}{\partial z} \tag{45}$$

$$\nabla_z f = (J_{f\circ g\circ h})^* \tag{46}$$

Il Calcolo di Wirtinger rende un po' piú semplice costruire un grafico computazionale per reti complesse aventi composizioni miste di operazioni olomorfe e non olomorfe.

3.5 Confronto con rete neurale

4 Ottimizzazione

4.1 ensemble di reti neurali

Spesso una rete di dimensioni finite non apprende completamente un particolare mapping o si generalizza male

Aumentare la dimensione o il numero di hidden laer il piú delle volte non porta a nessun miglioramento (Soulie, 1987).

Molti ricercatori hanno dimostrato che la semplice combinazione degli output di molti classificatori puó generare previsioni piú accurate di quelle di qualsiasi classificatore (Clemen 1989; Wolpert 1992). In particolare, la combinazione di reti neurali addestrate separatamente (comunemente indicato come ensemble di reti neurali) si é dimostrata particolarmente efficace (Alpaydin 1993; Drucker et al. 1994; Krogh e Vedelsby 1995; Maclin e Shavlik 1995; Perrone 1992).

Nella comunitá delle reti neurali, gli ensemble sono stati studiati da diversi autori come [1, 2, 3 di 1001]

Figura 1 illustra la struttura generale di un ensemble di reti neurali. Ogni rete nell'ensemble é per prima cosa addestrato e poi, per ogni esempio del training, l'output predetto di ognuna delle reti é combinato per produrre l'output dell'ensemble.

Le proprietá di un ensemble di reti neurali vegono esaminate e confrontate con le prestazioni di una singola rete.

Dopo il training, viene creato l'ensemble combinando gli output delle singole reti.

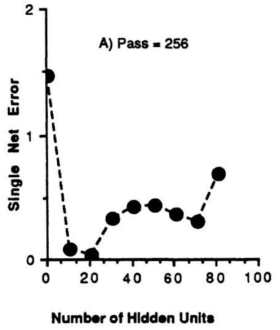
L'output del cluster puó essere utilizzato come ulteriore training restituendolo alle singole reti.

William P. Lincoln e Josef Skrzypek in $Synergyofclusteringmultiplebackpropagationnetworks$ confrontano le prestazioni di ogni singola rete con le prestazioni di un enseble di 5 reti neurali, con stessa struttura della singola.

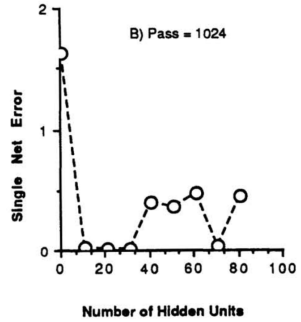
La miglior capacitá di apprendimento di un ensemble dipende dalla sinergia tra le singole reti e non sulle maggioi dimensioni rispetto alla singola rete.

L'errore di una singola rete dipende dalle dimensioni del hidden layer e dalla durata del training. Tuttavia in generale l'errore non é una funzione decrescente della dimensione dell'hidden layer. In figura 1 si mostra l'errore di una singola rete rispetto alla dimensione dell'hidden layer per diversi step di training. Invece la miglior capacitá di apprendimento di un ensemble dipende dalla sinergia tra le singole reti e non sulle maggioi dimensioni rispetto alla singola rete.

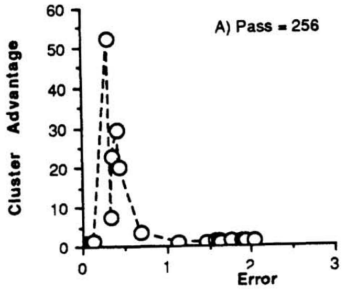
Vengono confrontati gli errori medi dei due sistemi. Una misura utile del vantaggio dell'ensemble si ottiene prendendo il rapporto tra l'errore di una singola rete e l'errore del judge. Questo rapporto sará inferiore o maggiore di 1 a seconda del valore assoluto dell'errore della singola rete neurale e dell'ensemble. Figura 1 mostra il vantaggio dell'ensemble rispetto agli errori della rete singola rispettivamente per 256 e 1024 step del training. Nei casi estremi in cui l'apprendimento sia nullo o quasi totale, l'ensemble non mostra particolari vantaggi, tuttavia si ha un notevole vantaggio se l'apprendimento é solo parziale



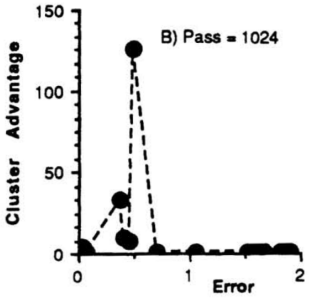
(a) da inserire



(b) da inserire



(a) da inserire



(b) da inserire

4.1.1 judge

Dato un singolo input, le reti neurali dell'ensemble daranno presumibilmente degli output Y_k diversi, che diventano l'input del *judge* che li combina per ottenere il risultato finale Y.

Esistono diversi metodi per calcolare Y, ma per semplicitá consideriamo i due casi piú semplici:

$$Y = \sum_{k=1}^{N} \frac{1}{N} Y_k \tag{47}$$

$$Y = \sum_{k=1}^{N} W_k Y_k \tag{48}$$

Il primo é dato da una semplice media dei Y_k, mentre nel secondo caso si considera una media pesata, con i W_k che indicano "l'affidabilitá" delle singole reti neurali. Seguendo questo ragionamento i W_k vengono di volta in volta modificati come segue:

$$W_k = W_k \cdot G \cdot \frac{e}{e_k} \tag{49}$$

$$e = \frac{1}{N} \sum k = 1^N e_k \tag{50}$$

$$e_k = |y - y_k| \tag{51}$$

Con e che indica il tasso di correzione e e_k é la deviazione dell'output della singola rete dall'output dell'ensemble

Dopo il periodo di training iniziale, si presume generalmente che la rete neurale non sia piú soggetta ad ulteriore training. Tuttavia, si puó restituire l'output del judge alle singole reti neurali come output desiderato. William P. Lincoln e Josef Skrzypek in $Synergyofclusteringmultiplebackpropagationnetworks$ affermano che questo procedimento migliori la resistenza al rumore e l'auto-organizzazione (*dacapirecosasia*).

4.1.2 ottimizzazione dell'ensemble

Precedenti lavori sia teorici (Hansen e Salamon 1990; Krogh e Vedelsby 1995) che empirici (Hashem et al. 1994; Maclin e shavik 1995) hanno dimostrato che un ensemble efficace dovrebbe consistere non solo in reti particolarmente performanti, ma anche in reti che commettano errori in parti distinte dello spazio degli input.

Una combinazione degli output di piú reti é utile solo se queste non concordano per qualsiasi input. Quantificando il disaccordo nell'ensemble risulta possibile esprimere questa intuizion rigorosamente.

Assumiamo che l'obiettivo sia quello di apprendere una funzione $f : R^N \to R$ per la quale abbiamo p coppie input-output per il training con $y_k = f(x_k)$ e $k = 1, \cdots, p$. L'ensenble consiste in N reti e chiamiamo $V^\alpha(x)$ l'output della rete α relativo all'input x. Definiamo nel modo seguente il valore ottenuto dal judge:

$$\overline{V}(x) = \sum_\alpha w_\alpha V^\alpha(x). \tag{52}$$

Consideriamo i pesi w_α come l'affidabilitá della rete α e di conseguenza li vincoliamo ad essere positivi e normalizzati, cioé $\sum_\alpha w_\alpha = 1$. L'*ambuiguitá* su un input x di un singolo membro dell'ensemble é definito come $a^\alpha(x) = \left(V^\alpha(x) - \overline{V}(x)\right)^2$. La *ambiguitá dell'ensemble* su un input x é data da:

$$\overline{a}(x) = \sum_\alpha w_\alpha a^\alpha(x) = \sum_\alpha w_\alpha \left(V^\alpha(x) - \overline{V}(x)\right)^2. \tag{53}$$

É quindi una varianza e misura il disaccordo delle reti sull'input x. L'errore quadratico della rete α e dell'ensemble sono rispettivamente:

$$e^\alpha(x) = \left(f(x) - V^\alpha(x)\right)^2 \tag{54}$$

$$e(x) = \left(f(x) - \overline{V}(x)\right)^2 \tag{55}$$

Aggiungendo e sottraendo $f(x)$ alla 53 si ottiene:

$$\overline{a}(x) = \sum_\alpha w_\alpha e^\alpha(x) - e(x). \tag{56}$$

Chiamando la media pesata degli errori delle singole reti $\overline{e}(x) = \sum_\alpha w_\alpha e^\alpha(x)$ allora la 55 diventa:

$$e(x) = \overline{e}(x) - \overline{a}(x). \tag{57}$$

Queste ultime formule possono essere mediate lungo la distribuzione degli input, ottenendo le seguenti:

$$E^{\alpha} = \int dx\, p(x)\; e^{\alpha}(x) \tag{58}$$

$$A^{\alpha} = \int dx\, p(x)\; a^{\alpha}(x) \tag{59}$$

$$E = \int dx\, p(x)\; e(x)\,. \tag{60}$$

Le prime due sono rispettivamente l'errore generalizzato e l'ambiguitá della rete α e E é l'errore generalizzato dell'ensemble. Dalla 57 otteniamo:

$$E = \overline{E} - \overline{A} \tag{61}$$

con $\overline{E} = \sum_{\alpha} w_{\alpha} E^{\alpha}$ la media pesata degli errori generalizzati delle singole reti e $\overline{A} = \sum_{\alpha} w_{\alpha} A^{\alpha}$ la media pesata delle ambiguitá delle singole reti. Questa equazione separa l'errore generalizzato in un termine che dipende dagli errori generalizzati delle singole reti e da un altro contenente le correlazioni tra le reti. Inoltre, il termine di correlazione A puó essere stimato interamente da dati non etichettati, non é richiesta alcuna conoscenza della funzione da approssimare. Il termine "senza etichetta" é tratto dai problemi di classificazione e in questo contesto si riferisce a un input x per il quale non si conosce il valore $f(X)$ della funzione target. Se l'ensemble é fortemente distorto, l'ambiguitá sará piccola, perché le reti implementano funzioni molto simili e concordano quindi gli input anche al di fuori del training. Pertanto l'errore generalizzato sará sostanzialmente uguale alla media degli errori delle singole reti. Se, d'altra parte, c'é una grande varianza, l'ambiguitá é alta e in questo caso l'errore di generalizzazione sará piú piccolo dell'errore di generalizzazione medio. Vediamo immediatamente ce l'errore generalizzato dell'ensemble é sempre minore della media pesata degli errori degli ensemble: $E < \overline{E}$. In particolare, per pesi uniformi:

$$E \leq \frac{1}{N} \sum_{\alpha} E^{\alpha} \tag{62}$$

Dalla 61 si ricava che aumentare l'efficienza dell'ensemble significa aumentare l'ambiguitá e quindi la discordanza tra le reti singole senza aumentarne ovviamente l'errore generalizzato. Come aumentare l'ambiguitá dell'ensemble? Un metodo puó essere quello di utilizare diverse tipologie di reti neurali, oppure si utilizzano set di training diversi. Inoltre, per essere in grado di stimare il primo termine in 61, sarebbe auspicabile una cross-validation. William P. Lincoln e Josef Skrzypek in $Synergyofclusteringmultiplebackpropagationnetworks$ testarono questo metodo per approssimare con un ensemble di reti neurali un'onda quadra in una variabile, mostrata in figura 3. Sono state utilizzate 5 reti con un hidden layer di 20 "neuroni", addestrate indipendenemente le une dalle altre mediande back-propagation utilizzando 200 esempi casuali. La "vera" generalizzazione e l'ambiguitá sono state stimate da un insieme di 1000 input e i pesi associati agli output delle singole reti per la combinazione erano uniformi $w_{\alpha} = 1/5$. In figura 4 viene mostrato l'errore generalizzato in funzione della dimensione K dei set per la cross-validation. Innanzitutto notiamo come l'errore generalizzato sia lo stesso per un set di cross-validation di dimensione 40 o 0. Tuttavia bisogna

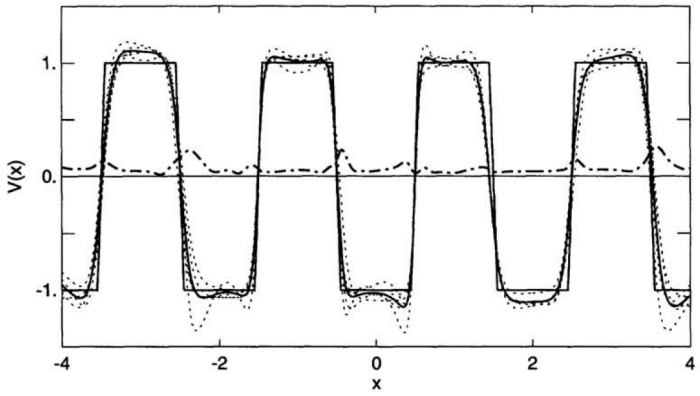


Figure 3: da inserire

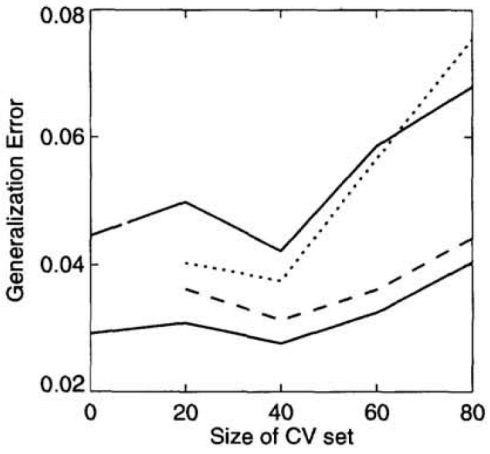


Figure 4: da inserire

considerare che sono stati scartati tutti i risultati di ensemble due o piú reti non convergenti.

La maggior parte dei lavori precedenti si é concentrata sulla combinazione degli output di reti piú performanti o ha indirizzato solo indirettamente il modo in cui generare un buon ensemble di reti, generando casualmente diverse topologie, impostazioni iniziali dei pesi, dei parametri o utilizzando solo una parte del set di training nella speranza di produrre reti che commettano errori per input distinti.

Per trovare direttamente un ensemble accurato e diversificato si possono sfruttare gli algoritmi genetici, creando una popolazione iniziale e utilizzando operatori genetici per creare continuamente nuove reti, mantenendo di volta in volta solo l'insieme di reti piú performanti e al contempo piú discordi tra loro.

In tabella 1 sintetizziamo l'agoritmo utilizzato. Viene creata una popolazione iniziale di reti neurali sottoposta a training. Di seguito si creano nuove reti

partendo dalle precedenti utilizzando operazioni "genetiche" come mutazioni e crossover. La nuova popolazione viene nuovamente addestrata, ponendo particolare attenzione ad utilizzare gli esempi classificati erroneamente dalla popolazione precedente. Viene quindi associato un punteggio determinato dalla funzione di fitness:

$$Fitness_i = Accuracy_i + \lambda Diversity_i = (1 - E_i) + \lambda D_i \tag{63}$$

con la diversitá definita come:

$$D_i = \sum \left[V_i(x) - \overline{V}(x) \right]^2. \tag{64}$$

Definiamo il termine di accuratezza come $A_i = 1 - E_i$ come accuratezza del set di convalida della rete e utilizziamo l'equazione 64 per calcolare il termine di diversitá D_i. Quindi normalizziamo separatamente ciascuno dei due termini. Normalizzare entrambi i termini permette a λ di mantenere o stesso significato attraverso i domini. Non essendo sempre chiero con quale valore si debba settare λ, solitamente ci si basa sulle seguenti regole. Il valore di λ non si modifica se l'errore dell'ensemble $\hat{E}$ diminuisce mentre consideriamo nuove reti. Cambia invece se si verifica una delle seguenti: (1) se l'errore della popolazione $\overline{E}$ non sta aumentando e la diversitá D sta diminuendo, aumentiamo λ; (2) se $\overline{E}$ sta aumentando e $\overline{D}$ non decresce, caliamo λ.

Tabella 1 **GOAL:** creare geneticamente un ensemble di reti accurate e diversificate.

- Creare e addestrare la popolazione iniziale di reti.
- Finché non si raggiunge un limite di efficacia o di tempo:
  - Utilizzare operazioni genetiche per creare nuove reti.
  - Addestrare le nuove reti utilizzando l'equazione 65.
  - Misurare la diversitá di ogni rete rispetto alla popolazione corrente (equazione 64).
  - Normalizzare i termini di accuratezza e diversitá delle singole reti.
  - Cacolare la funzione di fitness in equazione 63.
  - Aggiornare la popolazione utilizzando le N reti con valore maggiore per la 63.
  - Corregere λ dell'equazione 63
  - Utilizzare la popolazione attuale di reti come ensemble e combinare gli output delle reti seguendo l'equazione 49

Ribadiamo che una rete utile all'ensemble é quella che classifica correttamente i maggior numero possibile di casi, mentre pecca principalmente laddove le altre reti classificano correttamente. Ci preoccupiamo di questo aspetto durante la backpropagation moltiplicando la normale funzione di costo per un termine che misura l'errore combinato della popolazione su un determinato esempio:

$$Cost = \sum_{k \in T} \left| \frac{t(k) - \hat{o}(k)}{\hat{E}} \right|^{\frac{\lambda}{\lambda+1}} \left[t(k) - a(k) \right]^2 \tag{65}$$

dove $t(k)$ é il target e $a(k)$ é l'attivazione della rete per l'esempio k nel training set T. Si noti che, dato che la nostra rete non é ancora un membro dell'ensemble,

$\hat{o}(k)$ e $\hat{E}$ non dipendono dalla nostra rete; il nuovo termine é quindi una costante é perció una costante quando calcoliamo le derivate nella back propagation. Normalizziamo $t(k) - \hat{o}(k)$ dividendo per l'errore dell'ensemble $\hat{E}$ in modo che il valore medio del nostro nuovo termine sia circa 1 indipendentemente dalla correttezza dell'insieme. Ció é particolarmente importante con popolazioni particolarmente accurate, pioché $t(k) - \hat{o}(k)$ sará vicino a 0 per la maggior parte degli esempi e la rete verrebbe addestrata solo su un numero esiguo di esempi. L'esmpoenete $\frac{\lambda}{\lambda+1}$ rappresenta il rapporto di importanza del termine di diversitá nella funzione di fitness. Ad esempio, se λ é vicino a 0, la diversitá non é considereta importante e la rete viene addestrata con la consueta funzione di costo; tuttavia se λ é grande, la diversitá é cosiderata importante e il nuovo termine nella funzione di costo assume maggiore importanza. Combiniamo le previsioni delle reti prendendo una somma ponderata dell'output di ciascuna rete, dove ogni peso é definito come in precedenza. Riportiamo di seguito i risultati ottenuti da David W. Opitz e Jude W. Shavlik. L'algoritmo genetico che utilizziamo per generare nuove topologie di rete é l'algoritmo REGENT (Opitz e Shavlik, 1994). REGENT utilizza algoritmi genetici per effettuare ricerche nello spazio delle topologie della rete neurale (KN) basate sulla conoscenza. I KN sono reti le cui topologie sono determinate a seguitp della mappatura diretta di un insieme di regole di base che rappresentano ció che attualmente sappiamo del nostro compito. KBANN (Towell e Shavlik, 1994), per esempio, traduce un insieme di regole proposizionali in una rete neurale, quindi affina i pesi della rete risultante usando la back-propagation. Knn addestrati, come le reti KBANN, hanno dimostrato di generalizzare frequentemente meglio di molte altre tecniche di apprendimento induttivo come le reti neurali standard (Optiz 1995; Towell e Shavlik, 1994). L'uso di KNN ci consente di avere reti altamente corrette nel costro gruppo; tuttavia, poiché ogni rete nel nostro insieme é inizializzata con lo stesso insieme di regole specifiche del dominio, non ci aspettiamo che vi sia un grande disaccordo tra le reti. Un'alternatia che consideriamo nei nostri esperimenti é quella di generare casualmente la nostra opolazione iniziale di topologie di rete, poiché a volte le regole specifiche del dominio non sono disponibili. Abbiamo eseguito ADDEMUP sul set di problemi MAX di NYNEX e su tre problemi del progetto genoma umano che aiutano a localizzare i geni nelle sequenze di DNA. Ognuno di questi domini é accompagnato da una serie di regole approssimativamente corrette che descrivono ció che attualmente é noto sull'attivitá (Opitz 1995, Optiz e Shavlik 1994). I nostri esperimenti misurano l'errore del set di test di ADDEMUP su queste attivitá. Ogni ensemble é composto da 20 reti e gli algoritmi REGENT e ADDEMUP hanno considerato 250 reti durante la loro ricerca genetica. La tabella 1 presenta i risultati del caso in cui gli studenti creano casualmente la topologia delle loro reti. La prima riga della tabella 1, la migliore rete, deriva da una rete neurale a singolo strato in cui, per ogni fold abbiamo addestrato 20 reti contenenti tra 0 e 100 hidden nodes e usato un set di validazione per scegliere la migliore. La riga successiva, il bagging, contiene i risultati dell'esecuzione dell'algoritmo di bagging di Breiman (1994) su reti standard con un unico hidden layer, in cui il numero di hidden nodes viene impostato casualmente tra 0 e 100 per ogni rete. Il bagging é un "bootstrap", cioé un metodo dell'ensemble che forma ogni rete dell'ensemble con una diversa partizione dell'insieme di addestramento. Genera ogni partizione tracciando casualmente, con la sostituzione, N esempi dal set di addestramento, dove N

é la dimensione del set di addestramento. Breiman (1994) ha dimostrato che il bagging é efficace su algoritmi di apprendimento "instabili", come le reti neurali, in cui piccole cambiamenti nel set di addestramento comportano grandi cambiamenti nelle previsioni. La riga inferiore della tabella 1, ADDEMUP, contiene i risultati di una serie di ADDEMUP in cui la popolazione iniziale di 20 individui viene generata casualmente. I risultati mostrano che su questi domini la combinazione dell'output di piú reti addestrate di generalizza meglio rispetto al tentativo di scegliere la rete singola rete migliore. Mentre la tabella in altomostra la potenza degli insiemi di reti neurali, la tabella 2 mostra la capacitá di ADDEMUP di utilizzare le conoscenze a priori del problema. La prima riga della tabella 2 contiene i risultati di generalizzazione dell'algoritmo KBANN, mentre la riga successica, KBANN-bagging, contiene i risultati dell'ensemble in cui ogni singola rete dell'ensemble é la rete KBANN addestrata su una diversa partizione dell'insieme di addestramento. Sebbene ciascuna di queste reti inizi con la stessa topologia e impostazione di peso iniziali "grande" (i pesi risultanti dalla conoscenza specifica del dominio), piccoli cambiamenti nel set di allenamento producono ancora cambiamenti significativi nelle previsioni. Si noti inoltre che su tutti i set di dati, il bagging KBANN é buono o migliore dell'esecuzione del bagging su reti generate casualmente (bagging della tabella 1). La riga successiva, REGENT-combined, contiene i risultati della semplice combinazione, usando l'equazione 49, le reti nella popolazione finale di REGENT. ADDEMUP, l'ultima riga della tabella 2, differisce principalmente da REGENT-combined in due modi: (1) la sua funzione di fitness (equazione 63) tiene conto della diversitá piuttosto che della sola precisione della rete e (2) allena le nuove reti enfatizzando gli esempi errati dell'ensemble attuale. Pertanto, il confronto di ADDEMUP con REGENT-combined aiuta a testare direttamente la diversitá di ADDEMUP raggiungendo l'euristica, anche se i risultati aggiuntivi riportati in Opitz (1995) mostrano che ADDEMUP ottiene gran parte del suo miglioramento dalla sua funzione di fitness. Ci sono due ragioni principali per cui riteniamo che i risultati du ADDEMUP nella tabella 2 siano particolarmente incoraggianti: (1) confrontando ADDEMUP con REGENT-combined, testiamo esplicitamente la qualitá della nostra euristica e dimostriamo la loro fficacia e (2) ADDEMUP é in grado di utilizzare efficacemente le conoscenze di base per ridurre l'errore delle singole reti nel suo insieme, pur essendo in grado di creare abbastanza diversitá tra di loro in modo da migliorare la qualitá complessiva dell'ensemble.

Table 1: Standard neural networks (no domain-specific knowledge used)

	Promoters	Splice Junction	RBS	MAX
best-network	6.6%	7.8%	10.7%	37.0%
bagging	4.6%	4.5%	9.5%	35.7%
ADDEMUP	4.6%	4.9%	9.0%	34.9%

Table 2: Knowledge-based neural networks (domain-specific knowledge used)

	Promoters	Splice Junction	RBS	MAX
KBANN	6.2%	5.3%	9.4%	35.8%
KBANN-bagging	4.2%	4.5%	8.5%	35.6%
REGENT-combined	3.9%	3.9%	8.2%	35.6%
ADDEMUP	2.9%	3.6%	7.5%	34.7%

4.2 Funzione d'attivazione sticastica

4.3 regolarizzazione (forse)

5 Applicazione delle reti neurali complesse al MRF

5.1 introduzione al MRF

5.2 MRF con reti neurali a valori reali

5.3 MRF con reti neurali a valori complessi