

# The Forest CoverType dataset

Gian Lorenzo Meocci

17 settembre 2005

## Sommario

In questo articolo ci proponiamo di realizzare una rete neurale in grado di classificare (con la maggiore accuratezza possibile) la famiglia di alberi presenti in una data foresta. In sostanza partendo da una serie di dati (54) che caratterizzano un albero cercheremo di classificarlo tra sette famiglie possibili. Il nostro scopo è quello di ottenere risultati simili, se non migliori, a quelli ottenuti dall'Università del Colorado. Poiché lavoriamo con le reti neurali illustrare tutte le prove effettuate complicherebbe l'esposizione, ci limiteremo perciò ad analizzare solo alcune reti per illustrare il problema e le difficoltà che si sono incontrate.

## 1 Struttura del database

Il database che abbiamo utilizzato è in formato MAT cioè utilizzabile attraverso l'ambiente MatLab. Esso è una matrice composta da 581.012 righe e 55 colonne.

Ogni riga è una descrizione di un albero con i suoi 54 attributi (le prime 54 colonne) e la famiglia di appartenenza (l'ultima colonna). I dati sono raw, cioè non pretrattati, e sono stati acquisiti da una serie di rilevazioni fatte dal Dipartimento di scienze forestali del Colorado.

Le prime 10 colonne rappresentano varie grandezze (altezza sul livello del mare, illuminazione, caratteristiche del terreno, ecc.). Le successive 44 sono valori binari (0, 1) indicano la presenza o l'assenza di alcuni elementi. Infine l'ultima colonna è il nostro target che è un numero intero che assume valori compresi tra 1 a 7.

## 2 Il primo tentativo

Non essendo degli esperti in scienze forestali quello che possiamo inizialmente tentare è di prendere i dati così come sono ed utilizzarli per addestrare una rete neurale feed-forward composta inizialmente da un solo strato nascosto e da un solo neurone di uscita. Tentiamo (in pratica) di vedere il target come una funzione discreta e di scoprire fino a che punto è possibile fare regressione sui dati stessi.

Avendo una così grande quantità di dati, per evitare un “out of memory”, è opportuno lavorare su un campione di dati che sia equivalente, da un punto di vista statistico, al database iniziale.

```
load covtype.data;
j=randperm(size(covtype,1));
mini=covtype(j(1:50000),:);
```

La figura 1 mostra l'istogramma che si ottiene dal database completo, mentre la figura 2 mostra quello ridotto che abbiamo chiamato *mini*.

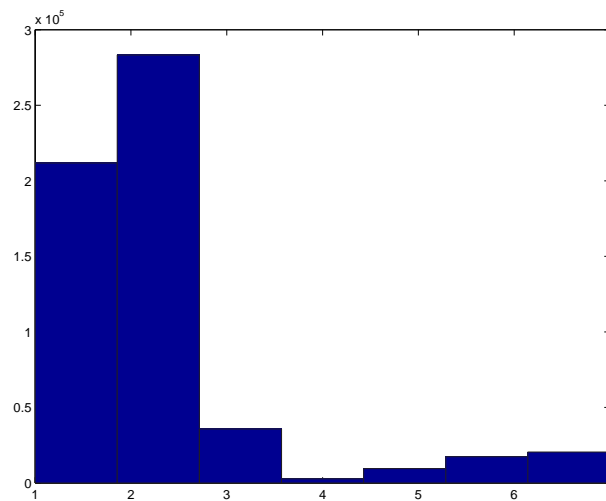


Figura 1: Distribuzione dell'intero dataset

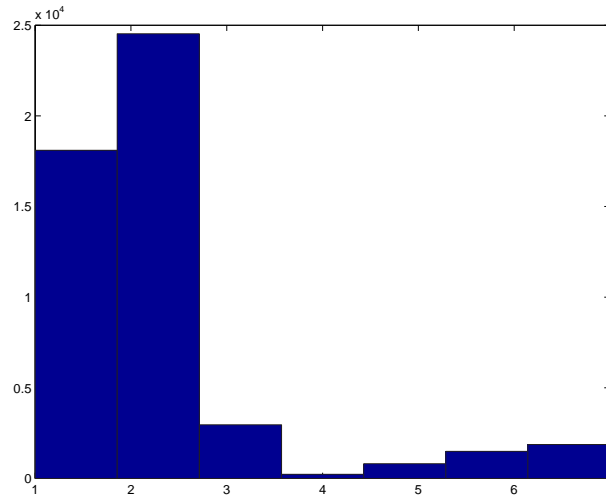


Figura 2: Distribuzione del dataset mini

L' unica differenza tra i grafici sopra riportati è rappresentata dalla scala di riferimento. Questo ci autorizza a lavorare in questo dataset ridotto, in quanto equivalente a quello completo.

Riporto adesso il codice MatLab per creare la rete neurale, che sarà lo scheletro principale che utilizzeremo per le nostre prove.

---

```

1 % covtype.m
2 tt=mini;
3 clear mytarget nout;
4 myindex=eye(7);
5 for i=1:10
6     tt(:,i)=tt(:,i)/max(tt(:,i));
7     tt(:,i)=tt(:,i)-min(tt(:,i));
8 end
9 myinput=tt(1:10000,1:54)';
10 test=tt(10000:50000,1:54)';
11 target=tt(1:10000,55)';
12 tt=tt(10000:50000,55)';
13
14 mm=minmax(myinput);

```

```

15 net=newff(mm,[35 1],{'tansig' 'purelin'},'trainrp');
16 net.trainParam.epochs = 500;
17 net.trainParam.goal=1e-1;
18 net1=train(net,myinput,target);
19 out=round(sim(net1,test));
20
21 k=0;
22 for i=1:size(tt,2)
23     if tt(i)==out(i)
24         k=k+1;
25     end
26 end
27
28 acc=k/size(tt,2)*100

```

---

Utilizzando due tecniche di apprendimento la *trainidx* e la *trainrp* otteniamo i seguenti risultati:

<i>trainidx</i>	46.556%
<i>trainrp</i>	48.291%

Da svariate prove sperimentali si è visto che le migliori performance (50%) si ottengono utilizzando 35 neuroni con un singolo strato nascosto.

Dobbiamo applicare alcune trasformazioni che ci consentano di migliorare la capacità predittiva della rete.

### 3 Trasformazione del dominio

Una possibile soluzione consiste ad'esempio nel pretrattare i dati. Infatti, utilizzare dati grezzi non è generalmente una buona soluzione, dal momento che il dominio attraverso il quale si descrive il nostro sistema incide sia sulla generazione dell'output sia sulle performance degli algoritmi di apprendimento. Ci focalizziamo pertanto sulla rappresentazione dei dati di input e decidiamo di scalare i primi 10 dati da 0 a 1.

Utilizzando la prima rete (sempre *trainidx* e 35 neuroni nello strato hidden) otteniamo un' accuratezza del 57.746%. Siamo dunque saliti (solamente pretrattando i dati) di un buon 10% anche se ancora

lontani dal nostro obiettivo finale ( $\sim 70\%$ ).

Non ci resta che cambiare l'architettura della rete tentando, attraverso prove sperimentali, di migliorare le performance in predizione. Utilizzando una rete di feedforward così definita:

```
net=newff(mm,[35 4 1],{'logsig' 'logsig' 'purelin'},'trainrp');
```

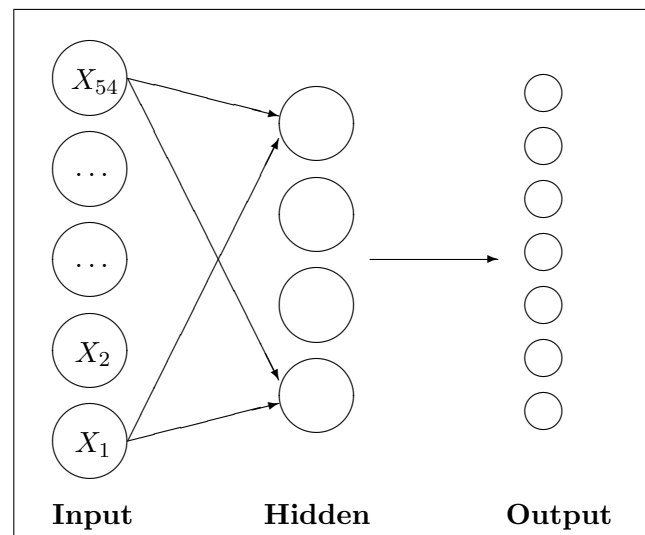
si è raggiunto un 63.753%, guadagnando un ulteriore 6% di accuratezza. Da questo punto in poi qualsiasi configurazione si possa provare non migliorerà le prestazioni.

Questo ragionamento si basa sull'esperienza e non su qualche teorema dimostrabile, dunque non escludo che non si possa migliorare ma ritengo che si debba cambiare ulteriormente l'approccio al problema.

Siccome trasformare il dominio del problema ha aumentato l'accuratezza della predizione, possiamo tentare di cambiare anche il codominio.

## 4 Trasformazione del codominio

L'idea di base consiste nel realizzare una uscita basata su 7 classificatori, ognuno per ogni possibile famiglia di alberi.



Utilizzando questa nuova rete neurale siamo riusciti ad ottenere le prestazioni desiderate. Per ottenere tale risultato si è dovuta modificare la struttura del programma, riadattando le fasi di creazione dell'input e dell' output.

---

```
1 tt=mini;clear mytarget nout;
2 myindex=eye(7); % matrice identita
3 for i=1:10
4     tt(:,i)=tt(:,i)/max(tt(:,i));
5     tt(:,i)=tt(:,i)-min(tt(:,i));
6 end
7
8 input=tt(1:10000,1:54)';
9 test=tt(10000:50000,1:54)';
10 target=tt(1:10000,55)';
11 vt=tt(10000:50000,55)';
12
13 for i=1:size(target,2)
14     mytarget(i,:)=myindex(target(i,:),:);
15 end
16 mytarget=mytarget';
17
18 mm=minmax(input);
19 net=newelm(mm,[57 7],{'tansig' 'logsig'},'trainrp');
20 net.trainParam.epochs=350;
21 net.trainParam.goal=1e-2;
22 net1=train(net,input,mytarget);
23 out=sim(net1,test);
24 out=round(out+0.01);
25 for i=1:size(out,2), nout(i)=matbin(out(:,i)');,end
26
27 k=0;
28 for i=1:size(vt,2)
29     if vt(i)==nout(i)
30         k=k+1;
31     end
32 end
33
34 acc=k/size(vt,2)*100
```

---

Sostanzialmente dobbiamo mappare i valori dell'attuale insieme  $\mathbf{Z}^7$  nel nuovo spazio  $\mathbf{Z}_7^2$  e dobbiamo successivamente riconvertirli per calcolare l'accuratezza del *test set*.

Per l'input non ci sono problemi, basta costruire la matrice identità  $\mathbf{I}_7$  (*myindex*) ed utilizzarla per costruire il nuovo target:

```
for i=1:size(target,2)
    mytarget(i,:)=myindex(target(i,:));
end
mytarget=mytarget';
```

Per l'output si è dovuto scrivere una funzione esterna (*matbin()*) in grado di mappare lo spazio calcolato della rete neurale con 7 neuroni di uscita, in quello originale, equivalente a quello generato dalla prima rete neurale presentata. Qui di seguito riporto la funzione sopra descritta

---

```
1 function d=matbin(x)
2 d=0;
3 for i=1:size(x,2)
4     if x(i)==1
5         d=i;
6     end
7 end
```

---

Dalle prove sperimentali è emerso che questa configurazione dà i migliori risultati, con un'accuratezza superiore al 72%.

Con questa serie di trasformazioni siamo riusciti a raggiungere (e superare) la meta premissa. Presentiamo per completezza anche il lavoro svolto con le SVM, in quanto è stato fondamentale per la realizzazione finale della rete neurale presentata.

## 5 Coverttype: Support Vector Machine

Durante lo studio del problema, e dopo svariati tentativi, si è deciso di utilizzare una SVM per capire ed analizzare ulteriormente il dataset iniziale. Per far ciò si è utilizzato il software SVMlight, utilizzandolo in due diverse modalità: regressione e classificazione (il ranking tende a consumare troppa memoria).

L'unico problema incontrato è stato quello di riportare il database dal formato Matlab a puro testo. Si è dunque scritto un programma in C++ per assolvere a tale compito. Il listato mostra tutti i classificatori utilizzati.

Utilizzando SVM è stato possibile studiare in maniera più dettagliata la struttura del dataset, e di conseguenza è stato più semplice capire come risolvere il problema. La prima prova effettuata, è stata quella di utilizzare SVM per fare regressione. Sin da subito è emerso che non si sarebbe potuto ottenere risultati soddisfacenti utilizzando reti neurali, dal momento che nessun kernel della nostra SVM è riuscito ha di superare il 50% d' accuratezza.

Un esperimento più significativo è stato quello di dividere il codominio in due insiemi:  $A=\{1,2\}$  e  $B=\{3,4,5,6,7\}$ . Questo nuovo classificatore è riuscito ha raggiungere il 75% di accuratezza, consentendoci di stimare la massima accuratezza raggiungibile da una nostra rete neurale. Questo upper bound ha senso dal momento che nessuna rete riuscirà a classificare meglio di una SVM dimensionata per riconoscere due superclassi (A,B) che rappresentano l'intero dataset.

Un ulteriore esperimento ha messo in evidenza che alcuni insiemi sono troppo piccoli per essere classificati con successo (si possono modellare come rumore), e che le performance complessive dipendono molto da come si riesce a dividere la prima dalla seconda classe.

## 6 Conclusione

L'utilizzo di reti neurali e SVM, può contribuire ad aumentare la conoscenza su un determinato set di dati, evidenziando alcune proprietà nascoste. Tuttavia il fattore umano e una buona fase di pre-processing sono ancora un elemento fondamentale nell'analisi dei dati.

Nel nostro problema i miglioramenti più significativi sono stati ottenuti trasformando i dati di input e modificando lo spazio di output, ma grazie alle SVM siamo riusciti a scoprire un limite superiore per l'accuratezza, e utilizzando una semplice rete neurale feedforward abbiamo infine costruito un sistema di predizione affidabile al 72%.

## 7 Codice

---

```
1 #include <iostream>
2 #include <fstream>
3 #define SPACE " "
4
5 using namespace std;
6
7 unsigned int get_double(char *ch){
8     int i=0,s=0,m=1;
9     unsigned int t=0;
10    for (i=0;i<4;i++){
11        s=(unsigned int)ch[i];
12        if (s<0)s=s+256;
13        t=t+m*s;
14        m=m*256;
15    }
16    return t;
17 }
18
19 int get_data(char *ch){
20     int t=0;
21     int s0,s1;
22     s0=(int)ch[0];
23     s1=(int)ch[1];
24     if (s0<0)s0=s0+256;
25     t=s0+256*s1;
26     return t;
27 }
28
29 void other_class();
30 void theother();
31
32 int *mat;
33 int row,col;
34
35 int main()
36 {
37     int i=0,j=0,k=0,n=0;
38     ifstream fid("c:\\lorenzo\\s.mat",ios::binary);
```

```

39     ofstream svm("c:\\lorenzo\\bigclass.txt");
40     ofstream sclass("c:\\lorenzo\\sclass.txt");
41     ofstream test("c:\\lorenzo\\test.txt");
42     ofstream test2("c:\\lorenzo\\test2.txt");
43
44     char ch[116]; fid.read(ch,116); // get descriptive text
45     fid.read(ch,8); //offset
46     fid.read(ch,2); // version
47     fid.read(ch,2);ch[2]=0; // endian indicator
48     cout<<ch<<endl;
49     fid.read(ch,4);ch[4]=0; // data type
50     int data_type=get_double(ch);
51     cout<<data_type<<endl;
52     fid.read(ch,4);ch[4]=0;
53     int number_byte=get_double(ch);
54     cout<<number_byte<<endl;
55     fid.read(ch,24);
56     fid.read(ch,4);ch[4]=0;
57     row=get_double(ch);
58     fid.read(ch,4);ch[4]=0;
59     col=get_double(ch);
60     cout<<"row_"<<row<<"_col_"<<col<<endl;
61     fid.read(ch,2); // nothing in this moment
62     fid.read(ch,2);ch[2]=0; // name of array
63     int name_size=(unsigned int)ch[0];
64     fid.read(ch,name_size);ch[name_size]=0;
65     string name=string(ch);
66     int jump=name_size*4-name_size;
67     fid.read(ch,jump+8);
68     mat=new int [col*row];
69
70     for (i=0;i<row*col;i++){
71         fid.read(ch,2);
72         n=get_data(ch);
73         mat[i]=n;
74     }
75
76     for (j=0,i=0;j<10000;j++){
77         if (mat[j+54*row]<3)svm<<"+1"<<SPACE;
78         else svm<<"-1_";

```

```

79     for ( i=0;i <54;i++)svm<<i+1<<" : "<<mat [ j+i*row]<<SPACE;
80     svm<<endl;
81     }
82
83     for ( i=0;j <25000;j++){
84     if (mat [ j+54*row] <3) test <<" +1" <<SPACE;
85     else test <<" -1_";
86     for ( i=0;i <54;i++)test <<i+1<<" : "<<mat [ j+i*row]<<SPACE;
87     test<<endl;
88     }
89
90     for ( j=0,i=0;j <10000;j++)
91     if (mat [ j+54*row] <3){
92     if (mat [ j+54*row]==1) sclass <<" -1" <<SPACE;
93     else sclass <<" +1_";
94     for ( i=0;i <54;i++)sclass <<i+1<<" : "<<mat [ j+i*row]<<SPACE;
95     sclass <<endl;
96     }
97
98     for ( i=0;j <25000;j++)
99     if (mat [ j+54*row] <3){
100    if (mat [ j+54*row]==1) test2 <<" -1" <<SPACE;
101    else test2 <<" +1_";
102    for ( i=0;i <54;i++)test2 <<i+1<<" : "<<mat [ j+i*row]<<SPACE;
103    test2 <<endl;
104    }
105
106    other_class ();
107
108    delete mat;
109    fid.close ();svm.close ();
110    sclass.close ();test.close ();
111    test2.close ();
112    system("PAUSE");
113    return 0;
114 }
115
116 void other_class ()
117 {
118     int i=0,j=0,k=0,f=0;

```

```

119     const string base="c:\\lorenzo\\";
120     const string fname[]={"s3.txt","s4.txt","s5.txt",
121                          / "s6.txt","s7.txt"};
122     const string ffname[]={"st3.txt","st4.txt","st5.txt",
123                          / "st6.txt","st7.txt"};
124     ofstream myfile [5], tfile [5];
125
126     for (k=0;k<5;k++){
127         cout<<" ";
128         myfile [k].open((base+fname [k]).c_str(),ios::out);
129         tfile [k].open((base+ffname [k]).c_str(),ios::out);
130
131         for (j=0,i=0;j <10000;j++)
132             if (mat [j+54*row]>2){
133                 if (mat [j+54*row]==k+3)myfile [k]<<" +1_";
134                 else if (j%2) myfile [k]<<" -1_";
135                 else continue;
136                 for (i=0;i <54;i++)myfile [k]<<i+1<<" : "<<mat [j+i*row]<<SPACE;
137                 myfile [k]<<endl;
138             }
139
140         for (i=0;j <25000;j++)
141             if (mat [j+54*row]>2){
142                 if (mat [j+54*row]==k+3){
143                     tfile [k]<<" +1_";
144                     myfile [k]<<" +1_";
145                     for (i=0;i <54;i++)myfile [k]<<i+1<<" : "<<mat [j+i*row]<<SPACE;
146                     myfile [k]<<endl;
147                 }
148                 else tfile [k]<<" -1_";
149                 for (i=0;i <54;i++)tfile [k]<<i+1<<" : "<<mat [j+i*row]<<SPACE;
150                 tfile [k]<<endl;
151             }
152
153         myfile [k].close();
154         tfile [k].close();
155         cout<<" ";
156     }
157     cout<<endl;
158 }

```

```

159
160 void theother()
161 {
162     int i=0,j=0;
163     ofstream myfile("c:\\lorenzo\\theother.txt");
164     ofstream tfile("c:\\lorenzo\\ttheother.txt");
165
166     for(j=0,i=0;j<10000;j++)
167     if(mat[j+54*row]>2){
168     myfile<<(float)1/mat[j+54*row]<<SPACE;
169     for(i=0;i<54;i++)myfile<<i+1<<" : "<<mat[j+i*row]<<SPACE;
170     myfile<<endl;
171     }
172
173     for(i=0;j<25000;j++)
174     if(mat[j+54*row]>2){
175     tfile<<(float)1/mat[j+54*row]<<SPACE;
176     for(i=0;i<54;i++)tfile<<i+1<<" : "<<mat[j+i*row]<<SPACE;
177     tfile<<endl;
178     }
179
180     myfile.close();
181     tfile.close();
182 }

```

---