Table of Contents

Titolo: Resoconto del modello sviluppato per la competizione beeViva 2018

Autore: Leonardo Padovan

Data: 16/04/2018

Abstract

L'oggetto della competizione è predire il prezzo delle bottiglie di vino conoscendone le informazioni d'etichetta (zona, uve, cantina etc) e le recensioni di un famoso sito del settore. Si porta quindi una soluzione al problema usando l'analisi semantica del testo con reti neurali ricorrenti e quindi reti neurali per tutti gli altri dati.

Introduzione

L'approccio scelto in corso di competizione per la risoluzione del problema è stato quello delle reti neurali. Il motivo sta nella loro adattabilità, flessibilità e robustezza; ovviamente a patto di saper ben condizionare il problema. Soprattutto durante l'allenamento col dataset della competizione 2017 si sono rivleate più affidabili e performanti rispetto ad altri metodi quali KNN, foreste di alberi decisionali, SVM etc.

Il linguaggio usato è invece Matlab, principalmente per via dell'ambiente di sviluppo che consente di maneggiare comandi e variabili con più facilità e immediatezza.

Purtropppo non ho avuto modo di fare un'analisi statistica dettagliata dei dati e ho preferito impostare subito la soluzione con reti neurali. I punti chiave sono la log-normalizzazione della variabile di uscita *price* e l'ordinamento e normalizzazione delle variabili di ingresso. Ho altresì intravisto fin da subito il vantaggio competitivo di eseguire un'analisi semantica delle recensioni, per cui ho speso le prime due ore a sviluppare un algoritmo in questa direzione. Purtroppo valutando quanto tempo mi sarebbe stato necessario per portarlo a termine ho preferito sospendere e sviluppare direttamente la rete neurale coi dati rimanenti per avere comunque un risultato da sottomettere. Ne ho sottomessi solo due, uno con rete neurale con funzione di attivazione ReLU e quello finale con funzione sigmoide (a tangente iperbolica). Fortuanatamente nonostante il punteggio parziale non fosse promettente, grazie alla buona capacità di

generalizzazione dell'algoritmo usato sono riuscito a ottenere un buon risultato sul punteggio finale (quasi coincidenti, da 34.47 a 33.57).

Purtroppo la soluzione portata non conteneva l'analisi semantica, che ho poi deciso di sviluppare comunque e che qui riporto (usando la nuova libreria di analisi testuale disponibile con Matlab 2018a, -versione che purtroppo non avevo durante la competizione-).

Inoltre durante lo sviluppo ho notato alcune peculiarità del dataset: la prima circa il fatto che alcune recensioni contenevano indicazioni sul prezzo e quindi la possibilità di apportare delle correzioni manuali. Non ho comunque proseguito su questa strada data la poca significatività dei campioni coinvolti (meno di 200 su oltre 70000) e il tempo che sarebbe stato necessario. La seconda invece, erroneamente sottovalutata, circa il fatto che alcune righe erano duplicate, e quindi si apriva la possibilità di ricopiare i prezzi dai campioni di training ai rispettivi duplicati nel set di test. Purtroppo in competizione non mi sono speso per quantificare il fenomeno che in realtà avrebbe permesso di ricavare circa un terzo delle soluzioni chieste.

Da notare anche come la presenza di costì tanti duplicati può portare a risultati più ottimistici rispetto a quanto si potrebbe riuscire a fare con dati unici.

Per dare una stima dei tempi l'analisi sentimentale nella versione qui riportata avrebbe richiesto (inluso l'apprendimento delle nuove librerie) almeno 20 ore aggiuntive. Probabilmente la versione più grezza che stavo approntando durante la competizione solo 2 o 3. La correzione grazie i duplicati almeno 2 ore aggiuntive, mentre la correzione con le indicazioni di prezzo solo mezz'ora in più, il tempo di leggere le recensioni e correggere manualmente i prezzi.

Qui di seguito riporto lo svolgimento di quanto fatto durante la competizione, delle riflessioni sulla rete neurale usata e quindi i miglioramenti che non ho potuto apportare durante il contest.

```
% Inizializzo l'ambiente di lavoro
clear; close all; clc;
```

Studio dei dati

In input ci vengono fornite 7 variabili categoriche: *country*, *province*, *region_1*, *region_2* (sottoregione, opzionale, i molti campi vuoti sono sostituiti da 'NA') *winery*, *variety*, *designation*; 1 variabile numerica: *review_score*; 1 campo di testo: *review*. In output ci viene invece fornita 1 variabile numerica: *price*.

```
% Carico le tavole dati
fprintf('\nLoading data...')
dataTrain = readtable('training_set.csv','Format','%C%C%C%C%C%C%C%q%f
%f');
dataTest=readtable('validation_set.csv','Format','%C%C%C%C%C%C%C%q
%f');
% Riordino le colonne ('review' diventa ultima prima di 'price')
dataTrain = dataTrain(:,[1,2,3,4,5,6,7,9,8,10]);
dataTest = dataTest(:,[1,2,3,4,5,6,7,9,8]);
fprintf(' Done\n')
% Preview della tabella di training
%disp( dataTrain(1:5,:) )
% Preview della tabella di test
%disp( dataTest(1:5,:) )
```

L'intuizione è che i prezzi seguano una distribuzione log-normale, dato che: non possono essere negativi, possono ammettere prezzi molto elevati ma saranno comunque concentrati attorno ad un prezzo medio.

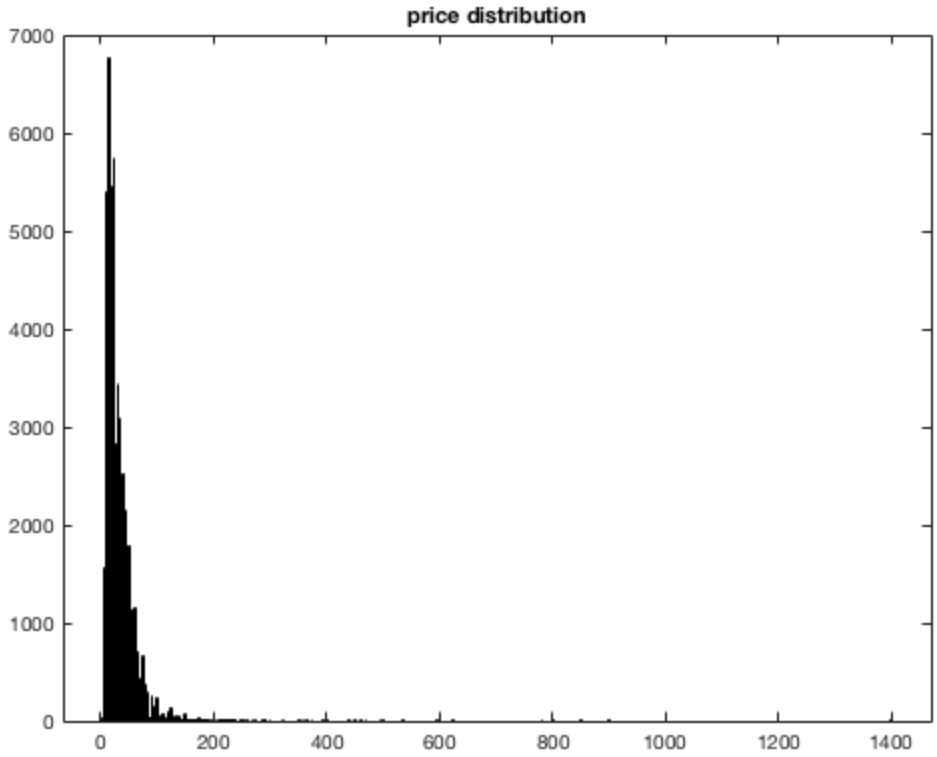
Estraggo quindi la distribuzione della variabile prezzo; al di là della scala (il prezzo più alto è 1400) si intuisce la forma della log-normale. Senza dover applicare procedimenti troppo laboriosi Matlab stesso mette a disposizione delle applicazioni dedicate (DistributionFitter), che comunque ho preferito non usare data la strettezza dei tempi.

```
fprintf('\nVisualizing data...\n')
% Controllo la distribuzione dei prezzi
figure(1)
histogram(dataTrain.price)
title('price distribution')
% Visualizzo box-plot e scatter-plot delle colonne di input contro il
 prezzo
%visualize(dataTrain,10,[1:8]) % ometto per brevità
```

Data la natura categorica dei dati in ingresso ci si pone il dubbio se i due set di training e test condividano le stesse categorie, i.e. che non ci siano categorie esclusive di un solo set. Flaggo quindi le colonne con categorie esclusive.

```
for i=1:7
    A=categories(dataTrain{:,i});
    B=categories(dataTest{:,i});
    res=length( setxor(A,B) );
    if (res~=0)
        fprintf('Categories not matching: column #%i\n',i)
    end
end
clear i A B res
```

```
Loading data... Done

Visualizing data...
Categories not matching: column #2
Categories not matching: column #3
Categories not matching: column #5
Categories not matching: column #6
Categories not matching: column #7
```



Preparazione dei dati

Avendo conferma della non omogeneità delle categorie dei due set, e sapendo come Matlab tratta gli array categorici, dovrò provvedere all'unione delle definizioni di categoria per le colonne imputate. Questa fase si rivelerà poi fondamentale per una migliore performance della rete neurale.

Un metodo molto veloce consiste nella semplice unione dei due set -facendo attenzione che quello di training in più contiene la colonna prezzo-.

```
fprintf('\nPreprocessing data...\n')
% Trattengo il numero di righe di ambo i set
m1=size(dataTrain,1);
m2=size(dataTest,1);
% Concateno le colonne di input e trattengo la colonna di output
data=[dataTrain(:,1:9);dataTest];
m=size(data,1);
price=dataTrain.price;
% Ridefinisco i due set di input
dataTrainIn=data(1:m1,:);
dataTestIn=data(m1+1:end,:);
```

Posso finalmente indagare sul numero di categorie per ogni colonna. Alcune saranno limitate (eg. *country*), altre invece avranno una granularità molto spinta (e.g. *designation* con 36337 categorie su 147459 elementi).

```
fprintf('Number of categories for each input column:\n\n')
```

```
% Estraggo i nomi delle colonne di input
vars=data.Properties.VariableNames;
% Elaboro il numero di categorie.
ncats=zeros(1,7);
for i=1:7
    ncats(i)=length(categories(data{:,i}));
end
%disp( array2table( ncats ,'VariableNames',vars(1:7)) )
disp( array2table( ncats(:,1:5) ,'VariableNames',vars(1:5)) )
disp( array2table( ncats(:,6:7) ,'VariableNames',vars(6:7)) )
clear i ncats
```

Ora si tratta di trasformare i dati categorici di input in forma numerica e costruire la matrice coi vettori da dare in pasto alla rete neurale. Saranno usate le prime 8 colonne tralasciando al momento la variabile *review*.

La ratio è quella di di assegnare un valore numerico ad ogni categoria. Sapendo il funzionamento intimo delle reti neurali, al fine di evitare la dispersione dei valori di attivazione sarà opportuno "ordinare" le categorie per garantire una sorta di "continuità" rispetto alla variabile *price*. Il metodo adottato è piuttosto semplice e veloce: per ogni colonna e per ogni categoria si è calcolato il prezzo medio per quella categoria e si è quindi usato questo risultato come criterio di ordinamento delle categorie. Per cui la categoria con valore 1 sarà rappresentata dai prezzi più bassi, viceversa per la categoria più alta.

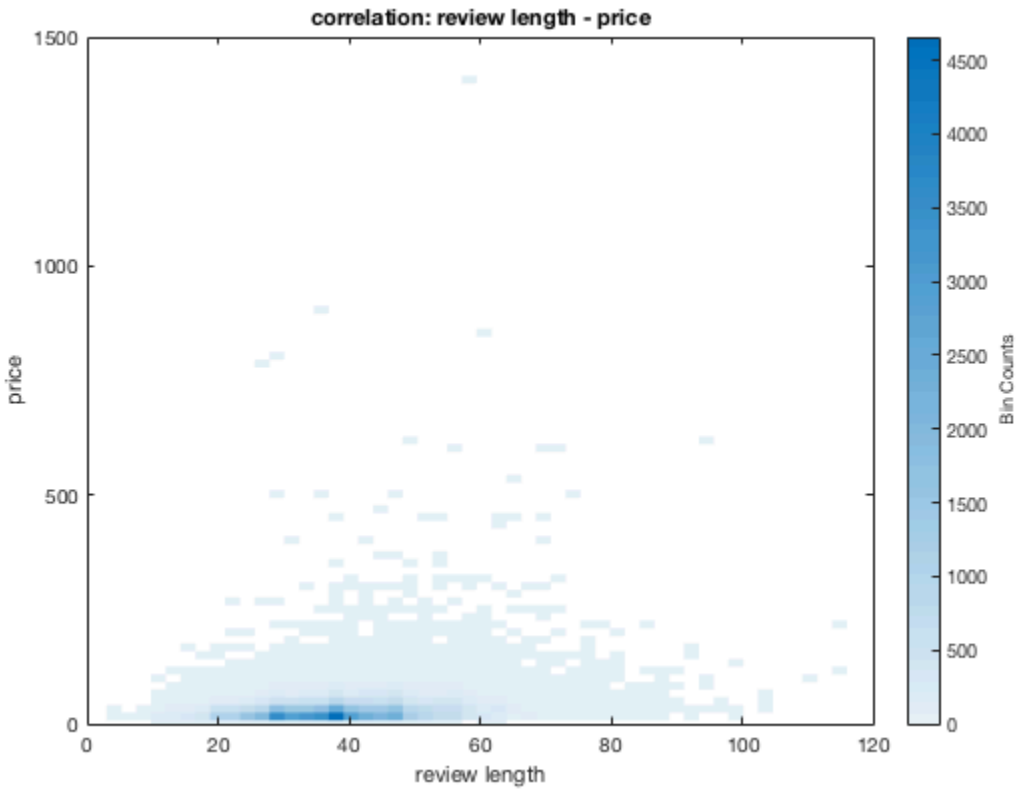
```
% Ottengo la matrice di input (train e test)
fprintf('Processing input matrix...')
% Il processo richiede tempo, per cui meglio caricare la tavola già
 pronta
%X=toMatrix(data,[price;zeros(m2,1)]);
load workspace/X.mat;
fprintf(' Done\n')
```

Avendo tralasciato la variabile *review* ho comunque deciso di provare a derivare una qualche informazione utile da aggiungere in ingresso. La scelta è ricaduta sulla lunghezza della recensione.

```
fprintf('Extracting reviews length...')
% Il processo richiede tempo, per cui meglio caricare la tavola già
 pronta
%{
nchar=zeros(m,1);
tic
parfor i=1:m
    str=data{i,9}{1};
    str=convertCharsToStrings(str);
    nchar(i)=length(strsplit(str));
end
toc
clear i str
%}
load workspace/nchar.mat;
X=[X,nchar];
fprintf(' Done\n\n')
% Metto a grafico la correlazione col prezzo
figure(2)
binscatter(nchar(1:m1),price); ylim([0,1500])
```

```
title('correlation: review length - price');
xlabel('review length');
ylabel('price')
```

```
Preprocessing data...
Number of categories for each input column:

    country    province    region_1    region_2    winery
    _______    ________    ________    ________    ______

       7          65          1204         19       11469

    variety    designation
    _______    ___________

      504          36337

Processing input matrix... Done
Extracting reviews length... Done
```



Normalizzazione

Passaggio cruciale è la normalizzazione dei dati. Ciò incrementerà la performance dell'algoritmo di training della rete neurale.

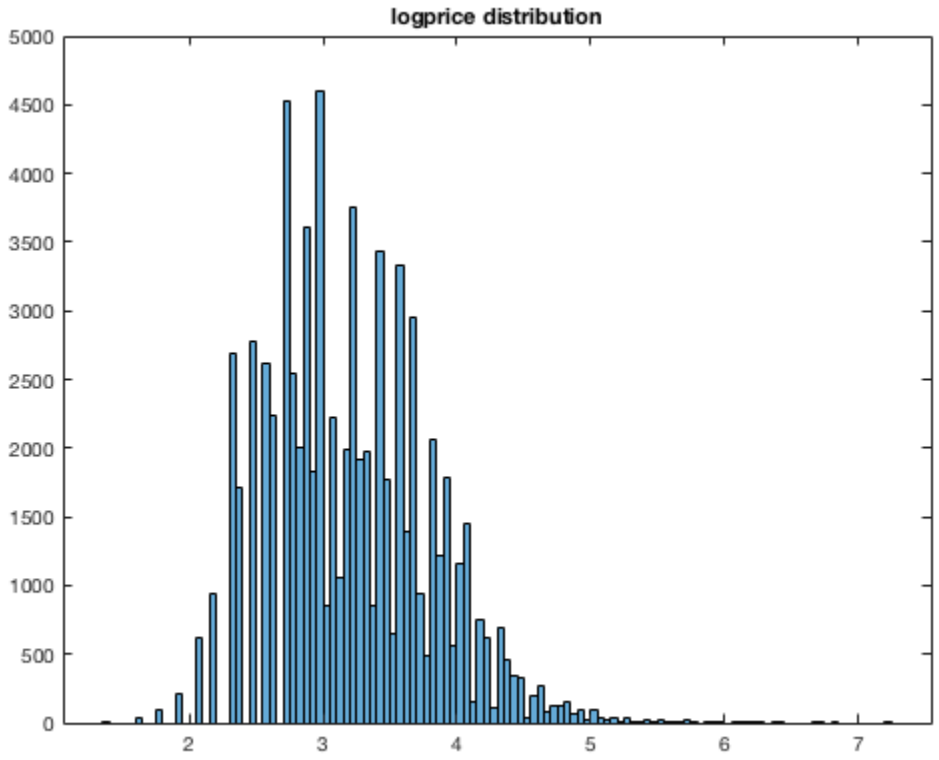
Un prima scelta era ricaduta su una riparametrizzazione min-max con minimo 0 e massimo 1 da accompagnare con una rete neurale con funzione di attivazione ReLU (data la positività dei dati e la caratteristica della funzione ReLU si poteva anche azzardare un training completamente privo di normalizzazione).

Alla fine in seguito ad alcuni dubbi circa la perdita di inforamzione sulle categorie riparametrizzate in 0 si è optato per una effettiva normalizzazione dei dati (media 0 e varianza 1) e una rete neurale con funzione di attivazione sigmoide

Nota a parte merita la riparametrizzazione della variabile di output. Come già notato *price* segue una distribuzione log-normale. Al fine di meglio condizionare l'algoritmo di apprendimento si vorrà ridistribuire questi valori nella maniera più uniforme possibile: nella ristrettezza dei tempi la scelta è ricaduta nell'utilizzare il semplice logaritmo dei prezzi, questo dovrebbe ridistribuire i dati secondo una curva normale, già più adatta allo scopo. Questa verrà poi normalizzata su media 0 e varianza 1.

```
fprintf('Normalizing data...')
% Ridistribuisco i prezzi secondo una normale
logprice=log(price);
figure(3)
histogram(logprice)
title('logprice distribution')
% Effettuo la normalizzazione.
sx=std(X,1); mx=mean(X,1); Xnorm=(X-mx)./(sx);
sy=std(logprice,1); my=mean(logprice,1); ynorm=(logprice-my)./(sy);
fprintf(' Done\n\n')
```

```
Normalizing data... Done
```



Training della rete neurale

La scelta della rete è ricaduta su una rete neurale di stampo classico con funzione di attivazione sigmoide a tangente iperbolico (da non confondere con la sigmoide logistica; sostanzialmente la stessa ma centrata sullo 0), non specificata perché di default. Il risultato finale sarà invece passato su un'attivazione lineare.

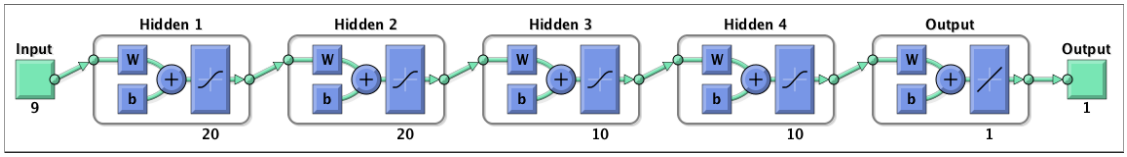
Il numero e dimensione dei layer è stato scelto un po' per esperienza, un po' per compromesso dato il tempo a disposione. In particolare non è stata effettuata nessuna ricerca dei parametri ottimali preferendo scelte a priori. Quindi 4 layer nascosti per permettere alla rete di definire comportamenti non-lineari sufficientemente complessi, 20 neuroni per i primi due (nella speranza di catturare utleriori features nella struttura dei dati oltre le 9 variabili di input) e poi 10 neuroni per gli ultimi due (nel timore di perdere troppo di definizione). Strutture più profonde o grandi avrebbero implicato tempi di training più lunghi (l'attuale ha impiegato diversi minuti sulla CPU di un laptop) oltre a ulteriori complessità (necessità di maggiore attenzione ai parametri di apprendimento della rete e quindi più tempo per il tuning). Va aggiunto che il comando usato permette di suddividere a sua volta il dataset utilizzato in subset di training e validazione (e test con altri algoritmi di apprendimento).

L'algoritmo di apprendimento è uno di quelli rivelatisi più robusti con l'esperienza (con backpropagation e regolarizzazione bayesiana dei pesi) con parametri di apprendimento di default (tra gli altri una velocità di apprendimento adattiva in risposta a diversi fattori interni). Tutti i parametri sono customizzabili, ma come già detto per velocità di esecuzione ho preferito rimanere sul default. L'unico strumento che mi sono riservato di usare è stato invece l'early stopping manuale del training della rete osservando i grafici di risposta (pendenza della rete di regressione sul subset di validazione e performance dell'errore quadratico medio MSE sui subset di training e validazione).

```
% Inizializzo la rete neurale
net=fitnet([20,20,10,10],'trainbr');
net.divideParam.trainRatio=.9;net.divideParam.valRatio=0;...
    net.divideParam.testRatio=.1;
%net.performParam.normalization = 'standard';
fprintf('Training the neural network...')
% Il training richiede tempo. Meglio caricare la rete già pronta
%[net,trainInfo]=train(net,Xnorm(1:m1,:)',ynorm');
load workspace/net02.mat
fprintf('Done\n')
view(net)
% Calcolo l'RMSE sul set di training
pred=net(Xnorm(1:m1,:)'); pred=pred.*(sy)'+my; pred=exp(pred);
fprintf('Training RMSE: %2.6f\n\n', sqrt( mean((pred'-price).^2) ))
% Generazione dei risultati finali
xtest=Xnorm(m1+1:end,:);
pred=net(xtest'); pred=pred.*(sy)'+my; pred=exp(pred);
% Scrittura su file di testo
fid=fopen('res02.txt','w');
fprintf(fid, '%f\n', pred');
fclose(fid);
```

E qui si chiude quanto fatto durante la competizione. I prossimi capitoli conterranno invece ulteriori passaggi non sviluppati per mancaza di tempo e che avrebbero sperabilmente portato ad un risultato migliore.

```
Training the neural network...Done
Training RMSE: 18.939384
```



Riflessioni sull'algoritmo finale

Durante la competizione come già detto si è scelto di log-normalizzare la variabile di output *price*. Nonostante ciò è rimasto il timore che qusta scelta potesse essere peggiorativa a causa dell'esponenziale da applicare ai risultati della rete neurale per riportarli alla loro forma originaria; esponenziale che potrebbe far esplodere gli errori (ossia piccole variazioni in uscita dalla rete verrebbero amplificate).

Due alternative contemplate erano allenare l'algoritmo usando la variabile *price* originale normalizzata (ma senza passare per il logaritmo) e la variabile originale depurata dagli outliers (la ratio è che mantenere i pochi outliers malcondizionerebbe l'algoritmo "tirando" a destra -verso i prezzi alti- le predizioni).

Qui ci si occupa di confrontare fianco a fianco le tre opzioni. Non avendo più a disposizione il set di validazione usato durante la competizione si dovrà provvedere a spezzare il set di training in un subset di training (75%) e uno di validazione (25%). Quindi si procede allenando la stessa struttura di rete neurale su 25 epoche coi tre output diversi, si calcolano i rispettivi errori sul set di training e validazione, si plottta la distribuzione e il grafico di regressione.

Si vede chiaramente come la scelta effettuata si sia rivelata ottimale. Inoltre emerge come il cap sui prezzi si sia comportato esattamente come da intuizione; probabilmente abbinare il cap alla log-normalizzazione potrebbe portare lievi benefici, ma non ho ritenuto opportuno indagare oltre.

```
net_insights(Xnorm,m1,price);
```

Un ulteriore punto di incertezza era la struttura della rete. Come già visto si è optato a priori per una rete con 4 layer nascosti con rispettivamente 20, 20, 10 e 10 neuroni. Inoltre per contenere i tempi si è proceduto con un early stopping a poche decine di epoche.

Quindi rimangono due aspetti da indagare: provare la performance di diverse strutture e valutare se un allenamento prolungato possa portare benefici. L'allenamento prolungato è stato condotto fino a 1000 epoche (o al raggiungimento dei parametri interni di goal). Le diverse reti contemplate sono invece una "large" (layers di dimensioni 20-20-10-10-10-5-5), una "mini" (10-10-10) e una "micro" (10-10). Da evidenziare anche i tempi di allenamento (su CPU a 2 cores):

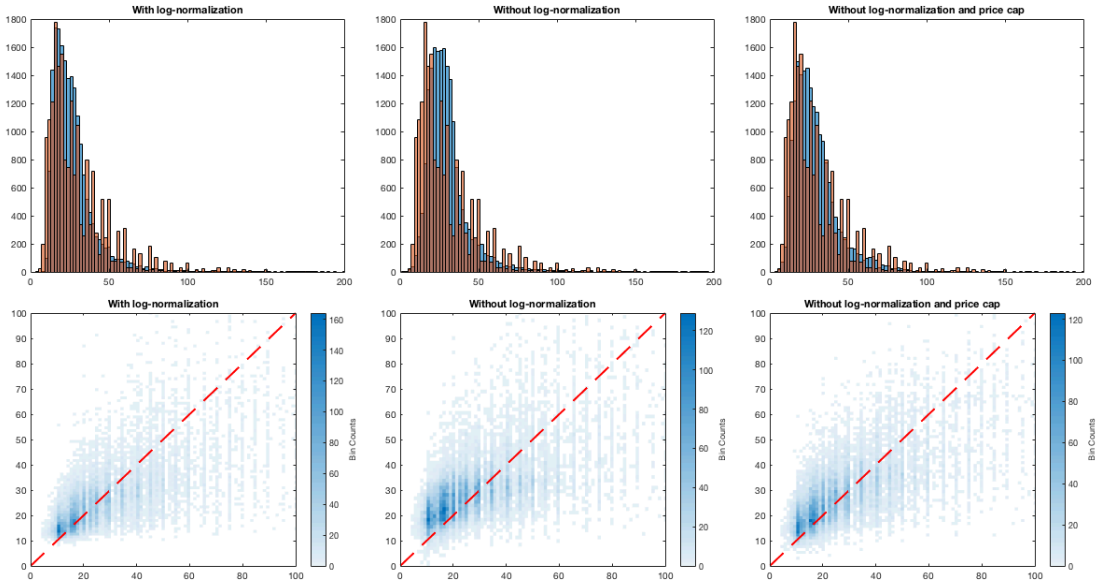
- rete "large": ~70 min (1000 epoche)
- rete "normale": ~50 min (1000 epoche)
- rete "mini": ~5 min (1000 epoche)
- rete "micro": ~1 min (278 epoche per raggiungimento del goal)

Probabilmente la rete scelta si è rivelata sovrabbondante rispetto alle necessità, dato che reti più snelle hanno ottenuto risultati analoghi in tempi decisamente minori. Rimarrebbe comunque da indagare sul numero di epoche ottimale (magari 25 potrebbe essere troppo poco e 1000 troppo). Comunque osservando i grafici di training generati da Matlab (qui non visibili) non c'è da aspettarsi miglioramenti rilevanti.

```
net_insights2(Xnorm,m1,price);
```

```
Network with log-normalization
Sub-Training RMSE: 19.391834
```

```
Sub-Validation RMSE: 20.160860

Network without log-normalization
Sub-Training RMSE: 18.874551
Sub-Validation RMSE: 20.383197

Network without log-normalization and price cap
Sub-Training RMSE: 19.971567
Sub-Validation RMSE: 20.487495

Large network
Sub-Training RMSE: 17.061356
Sub-Validation RMSE: 20.270873

Normal network
Sub-Training RMSE: 17.989535
Sub-Validation RMSE: 19.732756

Mini network
Sub-Training RMSE: 18.760367
Sub-Validation RMSE: 19.497855

Micro network
Sub-Training RMSE: 19.895233
Sub-Validation RMSE: 20.290689
```



Sentiment Analysis: analisi della variabile *review*

Il primo passo pensato per migliorare l'algoritmo di predizione è stato costruire un algoritmo semantico che processasse anche la variabile *review*. L'idea era che questo accorgimento avrebbe contraddistinto l'algoritmo dandogli un sensibile vantaggio rispetto tutti gli altri algorimi basati sulle sole variabili categoriche.

Ovviamente in fase di competiziome sarebbe stato difficile mettere in piedi un algoritmo semantico vero e proprio nel poco tempo a disposizione. Quindi l'approccio pensato sarebbe consistito nel leggere tutte le recensioni, depurarle della punteggiatura e di altri possibili token di disturbo, segmentare le parole e quindi costruire un vocabolario. Fatto qusto si trattava di estrarre degli "embeddings" da ogni singola recensione da poter usare poi come varibili di input per la rete neurale finale. Il metodo pensato per estrarre questi "embeddings" era piuttosto grezzo: per ogni recensione si costruiva un vettore one-hot di input della stessa lunghezza del vocabolario con valori zero o uno all'_i_-esima entrata e seconda che la *i*-esima parola del vocabolario comparisse nella recensione; e un vettore one-hot di uscita descrivente la fascia di prezzo (e.g. segmantando l'intervallo del logaritmo dei prezzi). Quindi si trattava di allenare una rete neurale che dalla recensione classificasse la classe di prezzo, il cui vettore di uscita avrebbe grossomodo rappresentato la probabilità della recensione di corrispondere ad ogni fascia di prezzo. Nel caso la lunghezza del vocabolario si fosse rivelata eccessiva si sarebbe potuto estrarre un sotto-vocabolario più ridotto (eventualemente scelto in maniera casuale).

Poco da dire, questa soluzione è stata investigata ed eseguita fino ad un certo punto (fino all'estrazione di un vocabolario di 43695 "vocaboli") e quindi abbandonata.

I principali ostacoli del metodo precedente sono innanzitutto l'estrazione del vocabolario. A scorrerlo si notavano molte parole "sporcate" da caratteri simbolici, numeri, errori di sintassi, abbreviazioni e soprattutto variazioni della stessa parola: di numero, di genere etc (e.g. per i nostri scopi sarebbe più dispersivo che utile distinguere tra *flaovur*, *favourous*, *flavourness* etc). Poi l'estrazione degli "embeddings" così tracciata sarebbe stata alquanto approssimativa ma soprattutto non si sarebbe sfruttato l'ordine temporale delle parole nella frase (certo si sarebbe potuto immettere i vettori one-hot in una rete neurale ricorrente ma così come sarebbero stati strutturati sarebbe stato proibitivo in termini di costo computazionale).

Per cui in questa seconda fase di revisione prendo in considerazione una soluzione più raffinata. Non userò il metodo ottimale che consisterebbe nell'utilizzo di mappe di embeddings già pronte (GloVe, Word2vec) ma mi cimenterò nel processo completo (sfruttando le librerie Matlab):

- estraggo tutte le recensioni;
- elimino la punteggiatura, porto tutto in caratteri minuscoli, rimuovo le "stop words" (articoli, avverbi, preposizioni etc), rimuovo le parole corte (<2 caratteri) e lunghe (>15 caratteri);
- normalizzo usando lo stemming di Porter (una sorta di riduzione alla radice delle parole)
- rimuovo le parole infrequenti (<10 apparizioni): vocabolario finale di 7999 parole;
- estraggo gli embeddings (sulla base del contesto) ottenendo per ogni vocabolo un vettore 50-dimensionale (dimostro anche come funzionano: dato il vettore di una parola, le parole coerenti nel contesto hanno vettori più "allineati", parole non coerenti hanno vettori più "perpendicolari");
- mappo le recensioni tramite gli embeddings, il risultato sarà una matrice 50x50 (50 vettori-parole 50-dimensionali). La scelta di una lunghezza massima di 50 è dovuta al voler mantenere l'algoritmo snello: la recensione più lunga ha 74 tokens e solo lo 0,19% supera i 50 tokens. Recensioni più lunghe vengono troncate alla fine, recensioni più corte vengono "riempite" con zeri all'inizio;
- ottenuti gli embeddings di input ricavo gli output spezzando il logaritmo del prezzo in 8 classi equispaziate. Volendo invece di una classificazione si potrebbe operare una regressione direttamente sul prezzo. A dir la verità queto approccio è riuscito a restituire un RMSE di training di 17.6675 e 21.9745 di validazione. La ragione per cui si è voluto procedere con la classificazione è quella di ottenere un'informazione più completa (la probabilità di appartenere ad ogni classe) da affinare con la rete finale.
- finisco allenando una rete neurale ricorrente (ad un layer LSTM) many-to-one. Nello specifico il modello è: input-lstm-dropout-dense-softmax-classification. Sono state tentate altre soluzioni (più layer, layer

bidirezionali etc) ma alla fine ho preferito rimanere col modello più semplice per via del costo computazionale (altrimenti sarei dovuto andare su GPU o su cloud e.g. Amazon EC2).

- allenata la rete ricalcolo tutti gli ouput per il training e il test della rete neurale finale.

Prima di allenare l'agoritmo semantico finale ho allenato una versione con dataset ridotto (i soliti sub-training e sub-validazione) per valutarne l'accuratezza (e l'accuratezza "estesa", ossia con uno scostamento consentito di una classe rispetto al valore atteso). Va fatto notare che con un training prolungato si sarebbe potuta ottenere una maggiore accuratezza ma mi sono fermato per motivi di tempo.

```
sentiments=sentiment(data,m1,price);
```

Finalmente la rete finale: costruisco la stessa rete usata durante la competizione ma ampliando gli input da 9 a 17 (9+8). I vettori usciti dall'algoritmo semantico sono già ristretti all'intervallo 0-1 (escono dal layer Softmax) per cui invece di normalizzarli mi limito a sottrarre 0.5.

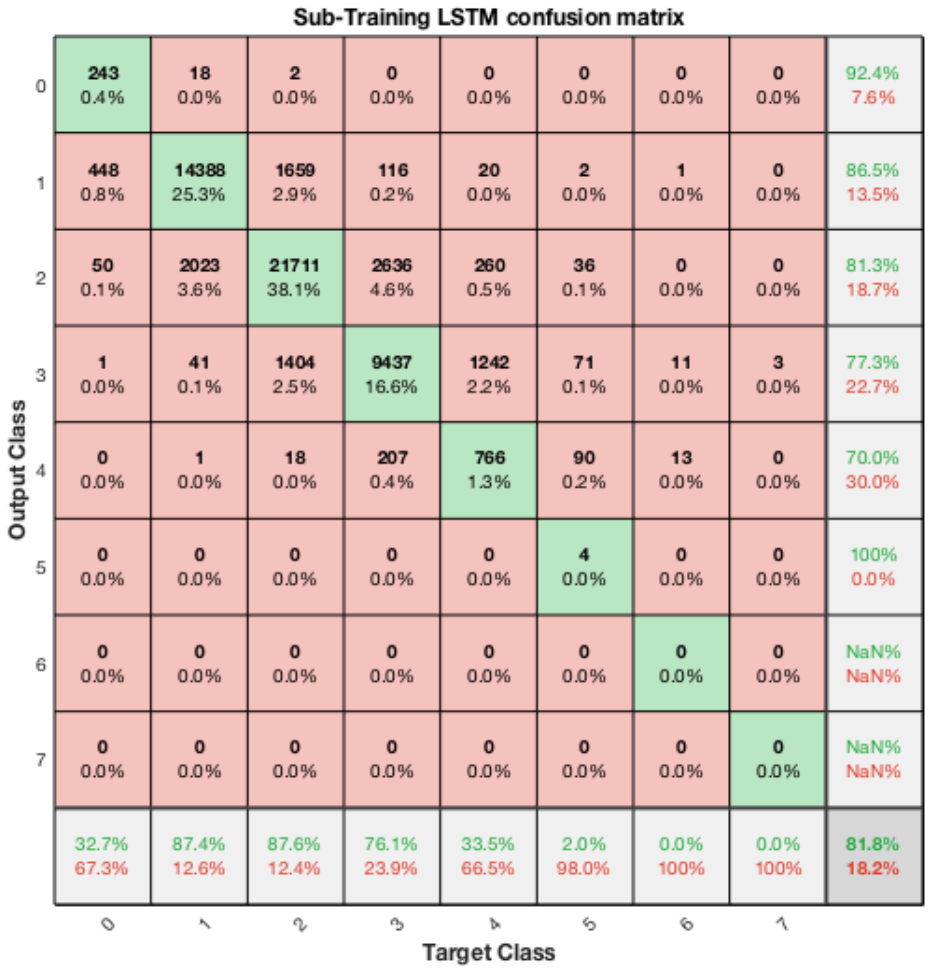
L'allenamento è starto eseguito per 100 epoche sul sub-set di training (~7 minuti) per avere l'RMSE di sub-training e sub-validazione; quindi per altre 50 epoche sull'intero set di training originale (~5 minuti).

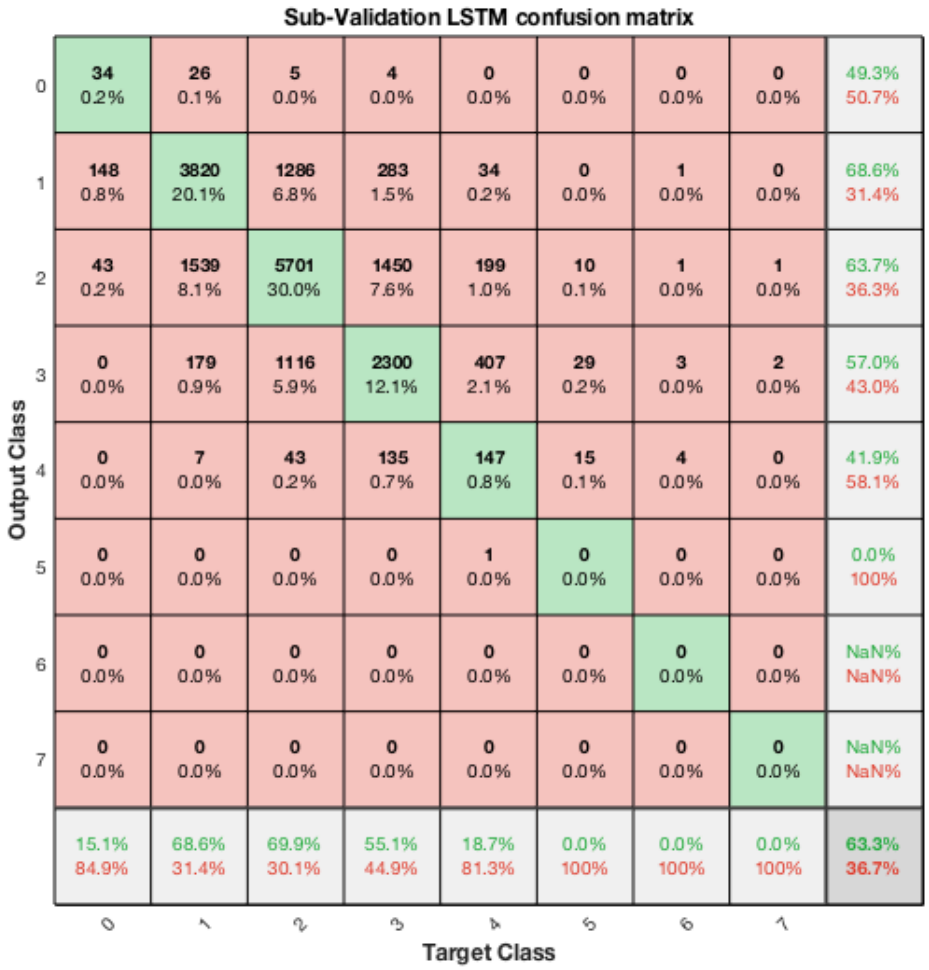
```
load workspace/cv.mat cv;
idtrain=training(cv); idval=test(cv);
Xfin=[Xnorm, sentiments-.5]; Xfint=Xfin(1:m1,:);

net=fitnet([20,20,10,10],'trainbr');
net.divideParam.trainRatio=.9;net.divideParam.valRatio=0;...
    net.divideParam.testRatio=.1;
%net.performParam.normalization = 'standard';
fprintf('Training the neural network...')
% Il training richiede tempo. Meglio caricare la rete già pronta
%[net,trainInfo]=train(net,Xfint(idtrain,:)',ynorm(idtrain)');
load workspace/netfin.mat
fprintf('Done\n')
% Calcolo l'RMSE sul set di training
pred=net(Xfint(idtrain,:)'); pred=pred.*(sy)'+my; pred=exp(pred);
fprintf('Training RMSE: %2.6f\n', sqrt( mean((pred'-
price(idtrain)).^2) ))
% Calcolo l'RMSE sul set di validazione
pred=net(Xfint(idval,:)'); pred=pred.*(sy)'+my; pred=exp(pred);
fprintf('Validation RMSE: %2.6f\n\n', sqrt( mean((pred'-
price(idval)).^2) ))

fprintf('Training the neural network on the whole dataset...')
% Il training richiede tempo. Meglio caricare la rete già pronta
%[net,trainInfo]=train(net,Xfin',ynorm');
load workspace/netfinex.mat
fprintf('Done\n')
% Calcolo l'RMSE sul set di training
pred=net(Xfint'); pred=pred.*(sy)'+my; pred=exp(pred);
fprintf('Training RMSE: %2.6f\n\n', sqrt( mean((pred'-price).^2) ))
% Generazione dei risultati finali
xtest=Xfin(m1+1:end,:);
pred=net(xtest'); pred=pred.*(sy)'+my; pred=exp(pred);
% Scrittura su file di testo
fid=fopen('res13.txt','w');
```

```
fprintf(fid, '%f\n', pred');
fclose(fid);
```

```
Loading tokenized reviews... Done
Details over bagOfWords
  WordCloudChart with properties:

           WordData: [1×7999 string]
           SizeData: [1×7999 double]
    MaxDisplayWords: 100

  Use GET to show all properties


Loading vocabulary embeddings... Done

Showing how embeddings work:
cabernet-red vector similarity (cosine): 0.594
cabernet-white vector similarity (cosine): 0.137

Loading reviews embeddings... Done

Classifing sentiments (train and validation subsets)...
Sub-Training accuracy: 0.8178
Sub-Training accuracy (extended): 0.9887
Sub-Validation accuracy: 0.6326
Sub-Validation accuracy (extended): 0.9553

Extracting ALL reviews sentiments... Done
Training the neural network...Done
Training RMSE: 14.256083
Validation RMSE: 18.516407

Training the neural network on the whole dataset...Done
Training RMSE: 14.940016
```


make
chardonnai
pinot
touch
sauvignon
vineyard
appl
note
good
tast
eleg
nice
raspberri
white
fine
smooth
firm
pear
show
light
fresh
tannin
miner
intens
charact
cabernet
red
mouthfeel
fruit
feel
drink
dark
syrah
tart
toast
green
soft
acid
finish
year
like
cola
clean
vanilla
bit
crisp
hint
berri
concentr
blend
dry
wine
oak
come
mouth
made
cherri
aroma
peach
tobacco
sweet
fruiti
dri
merlot
blackberri
flavor
just
currant
spice
balanc
rich
lemon
give
juici
herb
licoric
creami
black
palat
textur
ripe
pepper
ag
spici
delici
bottl
plum
full
nose
offer
well
chocol
structur
citru
open
bright
complex
long
tannic
there


Sub-Training LSTM confusion matrix
Output Class
Target Class
0 1 2 3 4 5 6 7
243 0.4% | 18 0.0% | 2 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 92.4% 7.6%
448 0.8% | 14388 25.3% | 1659 2.9% | 116 0.2% | 20 0.0% | 2 0.0% | 1 0.0% | 0 0.0% | 86.5% 13.5%
50 0.1% | 2023 3.6% | 21711 38.1% | 2636 4.6% | 260 0.5% | 36 0.1% | 0 0.0% | 0 0.0% | 81.3% 18.7%
1 0.0% | 41 0.1% | 1404 2.5% | 9437 16.6% | 1242 2.2% | 71 0.1% | 11 0.0% | 3 0.0% | 77.3% 22.7%
0 0.0% | 1 0.0% | 18 0.0% | 207 0.4% | 766 1.3% | 90 0.2% | 13 0.0% | 0 0.0% | 70.0% 30.0%
0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 4 0.0% | 0 0.0% | 0 0.0% | 100% 0.0%
0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN%
0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | NaN% NaN%
32.7% 67.3% | 87.4% 12.6% | 87.6% 12.4% | 76.1% 23.9% | 33.5% 66.5% | 2.0% 98.0% | 0.0% 100% | 0.0% 100% | 81.8% 18.2%



Riconoscimento di duplicati nel dataset

Avendo notato durante la visione dei dati alcune righe duplicate procedo con un'analisi più sistematica di tutti i duplicati, per quantificarli e per ricopiarne il prezzo tra set di training e set di test (prestando attenzione ad eventuali incoerenze).

```
% Ricavo la tabella coi prezzi aggiunti grazie alle ripetizioni
Xcorr=repetitions(X,price,m1,m2);

% Estrapolo indici e prezzi dalla tabella corretta
idtrain=Xcorr( Xcorr(:,11)<=m1, 11 );
idtest=Xcorr( Xcorr(:,11)>m1, 11 ) - m1;
pctrain=Xcorr( Xcorr(:,11)<=m1, 10 );
pctest=Xcorr( Xcorr(:,11)>m1, 10 );
idtest(pctest==0)=[]; pctest(pctest==0)=[];

fprintf('\nCorrecting contest network results...\n');
```

```
% Correggo i risultati della competizione
load workspace/net02.mat
% Calcolo l'RMSE sul set di training
pred=net(Xnorm(1:m1,:)'); pred=pred.*(sy)'+my; pred=exp(pred);
pred(idtrain)=pctrain;
fprintf('Training RMSE: %2.6f\n', sqrt( mean((pred'-price).^2) ))
% Generazione dei risultati finali
pred=net(Xnorm(m1+1:end,:)'); pred=pred.*(sy)'+my; pred=exp(pred);
% Calcolo del guadagno
score=33.57;
gain=sum((pred(idtest)'-pctest).^2)/m;
fprintf('Expeted Validation RMSE: %2.6f\n\n', sqrt(score^2-gain))
pred(idtest)=pctest; pred1=pred';
% Scrittura su file di testo
fid=fopen('res14.txt','w');
fprintf(fid, '%f\n', pred');
fclose(fid);

fprintf('Correcting final network results...\n');
% Correggo i risultati dell'algoritmo con analisi semantica
load workspace/netfinex.mat
% Calcolo l'RMSE sul set di training
pred=net(Xfint'); pred=pred.*(sy)'+my; pred=exp(pred);
pred(idtrain)=pctrain;
fprintf('Training RMSE: %2.6f\n\n', sqrt( mean((pred'-price).^2) ))
% Generazione dei risultati finali
xtest=Xfin(m1+1:end,:);
pred=net(xtest'); pred=pred.*(sy)'+my; pred=exp(pred);
pred(idtest)=pctest; pred2=pred';
% Scrittura su file di testo
fid=fopen('res15.txt','w');
fprintf(fid, '%f\n', pred');
fclose(fid);
```

Come si evince dai risultati il dataset contiene effettivamente un gran numero di campioni duplicati (circa la metà) e facendo riferimento al set di training sono riuscito a inferire quasi un terzo dei prezzi del set di test. Va fatto notare che tra tutti i duplicati alcuni portavano prezzi non coerenti tra loro. Questo è successo solo in 897 casi e di questi solo 95 avevano uno scostamento tra il prezzo più alto e quello più basso maggiore di 10. Data la marginalità del fenomeno si può anche tralasciare; in ogni caso laddove lo scostamento era poco (minore di 10 dollari o del 20% sul prezzo minore) ho deciso di inferire comunque il prezzo scegliendo quello medio.

Operando la correzione si nota un drastico miglioramento sul set di training. Sul set di test ho cercato invece di risalire al punteggio atteso conoscendo il punteggio ottenuto in competizione (assumendo che le sostituzioni siano tutte esatte).

```
Number of duplicated rows: 65422
Percentage of duplicated rows overall: 44.37%
Percentage of duplicated rows in train set: 34.17%
Percentage of duplicated rows in test set: 55.18%
Percentage of corrections in test set: 32.33%

Correcting contest network results...
```

```
Training RMSE: 10.673315
Expeted Validation RMSE: 32.039712

Correcting final network results...
Training RMSE: 9.139122
```

Cheat: reviews con indicazioni sul prezzo

L'ultimo accorgimento, scoperto durante la creazione del vocabolario, è che alcune recensioni contenevano indicazioni di prezzo:

- alcune esatte (e.g. *opaque purple and thick in the glass, this wine has inky, supercharged aromas of ripe blackberry and spice. it feels amped up but balanced, with more intensity and clarity than what most other $18 reds offer. the flavors of wild berry, mild herb and chocolate finish in a rich, toasty wrapping.*);
- altre approssimate (e.g. *for an under-$15 red blend, corte has it going on. it''s rich and full, almost syrupy on the nose, with lots of oak, ink and vanilla. shows power and packs a punch, with dark berry, chocolate and spicy flavors. a little tannic and scratchy, but with structure and framework to spare. malbec (80%), with bonarda and petit verdot.*);
- altre inutilizzabili (e.g. *oddly, this is preferable to el nido''s $140 cab-heavy fruit bomb. maybe that''s because monastrell is the go-to grape in jumilla, and this is 70% monastrell and only 30% cab compared to the opposite with el nido. all that said, this wine is ripe, approachable and dark, with excellent balance and a fudgy, warm finish of fruitcake and blackberry. drink now through 2013.*).

Le correzioni sono state quindi eseguite manualmente, anche azzardando alcuni casi dubbi. In ogni caso data l'esiguità del campione coinvolto (189 righe) non c'è da aspettarsi miglioramenti significativi sull'RMSE finale.

```
% Individuazione delle indicazioni di prezzo
cheat_idx=[];
tic
for i=1:size(dataTest,1)
    str=dataTest{i,9}{1};
    if ismember('$',str)
        cheat_idx=[cheat_idx;i];
    end
end
toc
% Controllo manuale dei cheat
%{
cheat=[array2table(cheat_idx),array2table(pred(cheat_idx)'),dataTest(cheat_idx,9)]
cheatcorr=[];
for i=1:length(cheat_idx)
    fprintf('Cheat # %i/%i\n',i,length(cheat_idx))
    temp=cheat{i,3}{1};
    a=1:100:length(temp); a=[a,length(temp)+1];
    for j=1:length(a)-1
        fprintf('%s\n',temp(a(j):a(j+1)-1))
    end
    fprintf('Predicted price: %4.2f\n',cheat{i,2})
    x=input('Price suggested? ');
    cheatcorr=[cheatcorr;x];
```

```
    fprintf('\n')
end
%}
load workspace/cheatcorr.mat cheatcorr
% Eseguo le correzioni
pred1(cheat_idx)=cheatcorr; pred2 (cheat_idx)=cheatcorr;
% Scrittura su file di testo
fid=fopen('res14.txt','w');
fprintf(fid, '%f\n', pred1);
fclose(fid);
fid=fopen('res15.txt','w');
fprintf(fid, '%f\n', pred2);
fclose(fid);
```

```
Elapsed time is 12.231309 seconds.
```

Published with MATLAB® R2018a