Softmax Classifier

Chiamiamo Softmax Classifier una rete neurale di classificazione in *N* categorie che all'output della rete applica la *softmax function*, interpretando poi i valori come la probabilità associata ad ogni categoria.

Consideriamo una rete neurale di classificazione in *N* categorie.
Sia *x* un input per la rete e $F(x)=(F_1(x),...,F_N(x))$ il corrispondente output.
Interpretiamo i valori $F_k(x)$ come il logaritmo della probabilità non normalizzata che *x* appartenga alla categoria *k* :

$$F_k(x)=\log P_k(x)$$

Allora la probabilità non normalizzata che *x* appartenga alla categoria *k* è:

$$P_k(x)=e^{F_k(x)}$$

mentre la probabilità normalizzata sarà data da:

$$\widetilde{P}_k(x)=\frac{e^{F_k(x)}}{\sum_{i=1}^{N} e^{F_i(x)}}$$

Tale funzione è la **softmax function**: applicandola all'output della rete possiamo poi interpretare i valori come la probabilità associata ad ogni categoria.

Cross-entropy loss function

Consideriamo una rete neurale di classificazione in *N* categorie (mutamente escludentesi).
Sia *x* un input per la rete, sia $y(x)=(y_1,...,y_N)$ il corrispondente output. Supponiamo che l'output sia interpretabile come la probabilità associata ad ogni categoria, ovvero che y_i rappresenti la probabilità che *x* appartenga alla categoria *i* .

Notiamo che $y(x)=(y_1,...,y_N)$ è di fatto la distribuzione di probabilità della variabile aleatoria discreta "categoria dell'oggetto *x*". Naturalmente si tratta di una distribuzione arbitraria determinata dalla configurazione dei parametri della rete.
Chiamiamo *ground-truth probability* la corretta distribuzione $\hat{y}(x)$; se *x* appartiene alla prima categoria, sarà:

$$\hat{y}(x)=(1,0,..,0)$$

e così via.

Allenare la rete significa minimizzare la differenza tra y e $\hat{y}$. Per farlo è naturalmente necessario misurare tale differenza. Esistono diverse funzioni che misurano la distanza $L(y,\hat{y})$ tra due distribuzioni di probabilità, una di queste è la **cross-entropy loss function**.

Si chiama **entropy** di una distribuzione di probabilità discreta $\hat{y}$, il numero:

$$H(\hat{y})=\sum_i \hat{y}_i \log\frac{1}{\hat{y}_i}=-\sum_i \hat{y}_i \log \hat{y}_i$$

Nel caso di una distribuzione del tipo $\hat{y}(x)=(1,0,..,0)$, si verifica facilmente che $H(\hat{y})=0$.

Si chiama **cross-entropy** tra la distribuzione di probabilità vera $\hat{y}$ di una variabile aleatoria e una distribuzione di probabilità stimata y il numero:

$$H(\hat{y},y)=\sum_i \hat{y}_i \log\frac{1}{y_i}=-\sum_i \hat{y}_i \log y_i$$

Si verifica che la *cross-entropy* è sempre maggiore della *entropy*.
Nel caso della *ground-probability*, ovvero di una distribuzione vera del tipo $\hat{y}(x)=(1,0,..,0)$, si ha:

$$H(\hat{y},y)=-\sum_i \hat{y}_i \log y_i=-\log(y_k)$$

dove *k* è l'unico indice tale che $\hat{y}_k\neq 0$ (e naturalmente $\hat{y}_k=1$).

Si chiama **divergenza di Kullback-Leipler** tra la distribuzione di probabilità vera $\hat{y}$ di una variabile aleatoria e una distribuzione di probabilità stimata y , la differenza tra *cross entropy* delle due distribuzioni e *entropy* della distribuzione vera:

$$D_{KL}(\hat{y}\|y)=H(\hat{y},y)-H(y)$$

e dunque:

$$D_{KL}(\hat{y}\|y)=\sum_i \hat{y}_i \log\frac{1}{y_i}-\sum_i \hat{y}_i \log\frac{1}{\hat{y}_i}=\sum_i \hat{y}_i \log\frac{\hat{y}_i}{y_i}$$

Minimizzare la divergenza di Kullback-Leipler tra due distribuzioni significa minimizzare la differenza tra la *cross-entropy* e l'*entropy*, e dunque minimizzare la distanza tra le due distribuzioni.
Poiché, come visto prima, nel caso della *ground-probability* si ha $H(\hat{y})=0$, si ha:

$$D_{KL}(\hat{y}\|y)=H(\hat{y},y)$$

e dunque minimizzare la divergenza di KL equivale a minimizzare la *cross-entropy*.

Support Vector Machine (SVM)

Chiamiamo *Support Vector Machine (SVM)* una rete neurale di classificazione in *N* categorie, il cui allenamento avviene tramite la *SVM loss function*.

Sia *x* un input per la rete e $y(x)=(y_1(x),...,y_N(x))$ il corrispondente output.
Supponiamo che *x* appartenga alla categoria *k* ($1\leq k\leq N$).
Misuriamo l'errore della rete come:

$$L=\sum_{i\neq k} max(0, y_i-y_k+\Delta)$$

dove Δ è un parametro arbitrario; tale funzione è detta ***SVM loss*** o anche ***hinge loss***.
Minimizzando la *SVM loss* la rete tenderà ad assegnare alla componente corretta dell'output $y_k(x)$ un valore maggiore rispetto a quelle scorrette, di una quantità Δ . Per convincersene è sufficiente notare che il valore della *hinge loss* è sempre non negativo, ed è uguale a zero se e solo se la componente corretta è maggiore di tutte le altre di un quantità maggiore o uguale a Δ .

Talvolta la *hinge loss* viene sostituita dalla *squared hinge loss*:

$$L=\sum_{i\neq k} max\left(0, y_i - y_k + \Delta\right)^2$$

che penalizza maggiormente eventuali errori (ne fa il quadrato).

Regolarizzazione (classificatori lineari)

Consideriamo un classificatore lineare in *N* categorie, ovvero una rete neurale con una funzione di output lineare; se $x=(x_1,\dots,x_L)$ è un input della rete, allora la funzione di output può essere rappresentata come:

$$y(x,W)=W\,x^T$$

dove *W* è una matrice $N\times L$ reale, contenente tutti i parametri della rete.

Supponiamo che la *loss function* della rete sia la *hinge loss:*

$$L[\,y(x,W)]=\sum_{i\neq k} max(0, y_i - y_k + \Delta)$$

dove *k* è la categoria cui appartiene *x*.
Ricordiamo che il valore della *hinge loss* è sempre non negativo, ed è uguale a zero se e solo se la componente corretta è maggiore di tutte le altre di un quantità maggiore o uguale a Δ . Segue che, fissato un *x* , esisteranno diverse configurazioni dei parametri, ovvero diverse matrici *W* , per cui $L[\,y(x,W)]=0$.

La *hinge loss* era solo un esempio: altre *loss function* presentano la stessa ambiguità, non determinando univocamente la configurazione dei parametri mediante la richiesta di minimizzazione.
È naturale chiedersi se tra le configurazioni dei parametri *W* che rendono minima la *loss function* ce ne sono di preferibili. L'esperienza ha suggerito i seguenti criteri di preferenza.
Sono preferibili matrici i cui elementi sono i più piccoli possibili (per evitare complessità numerica inutile).
Sono inoltre preferibili matrici i cui elementi sono il meno diversi possibile; ad esempio a $w=(1,0)$ preferiamo $w=(0.5,0.5)$. Ciò limita i fenomeni di *overfitting*.

Per fare in modo che la rete tenda verso la configurazione di parametri *W* preferibile si aggiunge alla *loss function* un termine che penalizza le matrici che con elementi molto grandi; spesso si usa la norma L^2 :

$$R(W)=\sum_i \sum_j W_{ij}^2$$

dove la sommatoria penalizza le matrici con elementi in generale grandi, il quadrato penalizza le matrici con qualche elemento molto grande. Per penalizzare si intende che la *loss function* è maggiore.

Dunque in generale scriviamo la *loss function* regolarizzata come la somma di due termini, uno detto *data loss* e l'altro *regularization loss*:

$$L[\,y(x,W)]=L_{data}[\,y(x,W)]+\lambda\,R(W)$$

dove λ è un parametro determinato empiricamente.