0.1 SVM

Le macchine a vettori di supporto (support-vector machines) sono dei modelli di apprendimento supervisionato associati ad algoritmi di apprendimento per la regressione e la classificazione. Dato un insieme di esempi per l'addestramento, ognuno dei quali etichettato con la classe di appartenenza fra le due possibili classi, un algoritmo di addestramento per le SVM costruisce un modello che assegna i nuovi esempi a una delle due classi, ottenendo quindi un classificatore lineare binario non probabilistico. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi appartenenti alle due diverse categorie siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono quindi mappati nello stesso spazio e la predizione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricade.

Oltre alla classificazione lineare è possibile fare uso delle SVM per svolgere efficacemente la classificazione non lineare utilizzando il metodo kernel, mappando implicitamente i loro ingressi in uno spazio delle caratteristiche multi-dimensionale.

In questo lavoro abbiamo utilizzato *SVC* di *sckit-learn* e siamo andati a selezionare i parametri che più influivano sui risultati e solo dopo siamo andati a ricercare il valore migliore per essi (tuning).

0.1.1 Tuning degli iperparametri

Nei modelli di Machine Learning ci sono alcuni parametri, noti come Hyperparameters, che non possono essere appresi direttamente dai dati. Sono comunemente scelti dagli esseri umani sulla base di qualche intuizione o prova. Questi parametri mostrano la loro importanza migliorando le prestazioni del modello come la sua complessità o il suo tasso di apprendimento. I modelli possono avere molti iperparametri e trovare la migliore combinazione di parametri può essere trattato come un problema di ricerca.

SVM ha anche alcuni iperparametri (come i valori C o gamma da usare) e trovare l'iperparametro ottimale è un compito molto difficile da risolvere. Ma può essere trovato semplicemente provando tutte le combinazioni e vedere quali parametri funzionano meglio. L'idea principale alla base è quella di creare una griglia di iperparametri e provare tutte le loro combinazioni.

GridSearch

Scikit-learn ha questa funzionalità integrata con *GridSearchCV*. GridSearchCV prende un dizionario che descrive i parametri che potrebbero essere provati su un modello per

addestrarlo. La griglia dei parametri è definita come un dizionario, dove le chiavi sono i parametri e i valori sono le impostazioni da testare.
Ci siamo soffermati sui tre parametri che abbiamo ritenuto più importanti : *C, kernel e gamma.*
Infine abbiamo scelto come kernel quello lineare e lasciato perdere il gamma (tenendo quello di default) data la natura del problema, e siamo quindi andati alla ricerca dell'iperparametro ottimo **C** utilizzando il dataset bilanciato (vedi sezione **??**).

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
# defining parameter range
param_grid = {'C': np.logspace(-3, 2, 6), 'kernel': ['linear']}
svm_estimator = svm.SVC()
grid = GridSearchCV(svm_estimator, param_grid, refit = True, verbose =
3, n_jobs=-1)
# fitting the model for grid search
grid.fit(X_train, y_train)
```

Come si può vedere dal codice il range di $C = \{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$ e questo è stato scelto in base a delle ricerche nella letteratura.

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits

[CV 1/5] END .....C=0.001, kernel=linear;, score=0.633 total time= 6.1min
[CV 3/5] END .....C=0.001, kernel=linear;, score=0.612 total time= 8.0min
[CV 4/5] END .....C=0.001, kernel=linear;, score=0.611 total time= 9.2min
[CV 2/5] END .....C=0.001, kernel=linear;, score=0.640 total time=10.2min
[CV 5/5] END .....C=0.001, kernel=linear;, score=0.589 total time= 8.9min
[CV 1/5] END ......C=0.01, kernel=linear;, score=0.639 total time=10.7min
[CV 2/5] END ......C=0.01, kernel=linear;, score=0.660 total time=13.0min
[CV 3/5] END ......C=0.01, kernel=linear;, score=0.627 total time=13.3min
[CV 4/5] END ......C=0.01, kernel=linear;, score=0.607 total time=13.7min
[CV 2/5] END .......C=0.1, kernel=linear;, score=0.652 total time=16.6min
[CV 5/5] END ......C=0.01, kernel=linear;, score=0.601 total time=23.7min
[CV 3/5] END .......C=0.1, kernel=linear;, score=0.618 total time=15.2min
[CV 1/5] END .......C=0.1, kernel=linear;, score=0.628 total time=23.2min
[CV 4/5] END .......C=0.1, kernel=linear;, score=0.604 total time=14.8min
[CV 5/5] END .......C=0.1, kernel=linear;, score=0.602 total time=17.0min
[CV 1/5] END .......C=1.0, kernel=linear;, score=0.612 total time=17.2min
[CV 2/5] END .......C=1.0, kernel=linear;, score=0.666 total time=25.4min
[CV 3/5] END .......C=1.0, kernel=linear;, score=0.624 total time=17.4min
[CV 4/5] END .......C=1.0, kernel=linear;, score=0.608 total time=15.2min
[CV 5/5] END .......C=1.0, kernel=linear;, score=0.611 total time=14.2min
[CV 3/5] END ......C=10.0, kernel=linear;, score=0.624 total time=13.2min
[CV 1/5] END ......C=10.0, kernel=linear;, score=0.624 total time=17.1min
[CV 2/5] END ......C=10.0, kernel=linear;, score=0.651 total time=27.1min
[CV 5/5] END ......C=10.0, kernel=linear;, score=0.607 total time=14.7min
[CV 4/5] END ......C=10.0, kernel=linear;, score=0.616 total time=29.5min
[CV 1/5] END .....C=100.0, kernel=linear;, score=0.615 total time=20.0min
[CV 4/5] END .....C=100.0, kernel=linear;, score=0.608 total time= 9.8min
[CV 2/5] END .....C=100.0, kernel=linear;, score=0.643 total time=15.5min
[CV 3/5] END .....C=100.0, kernel=linear;, score=0.622 total time=14.3min
[CV 5/5] END .....C=100.0, kernel=linear;, score=0.605 total time=12.4min

Best params : {C=0.01, kernel='linear'}
```

E quindi è stato scelto $C = 0.01$.