Machine Learning
K-Nearest Neighbours

Francesco Pugliese, PhD

neural1977@gmail.com

Introduzione

- ✓ Gli algoritmi di Machine Learning possono essere divisi in modelli parametrici e non-parametrici.

- ✓ Sfruttando i modelli parametrici siamo capaci di stimare i parametri del modello a partire dal Training Set al fine di ottenere una funzione capace di classificare nuovi dati senza ricorrere più al Training Set originale. Tipici esempi dei modelli parametrici sono il perceptrone, la regressione logistica o la **Support Vector Machine** lineare (**SVM**).

- ✓ Dall'altro versante, i modelli non-parametrici sono caratterizzati da un numero prefissato di parametri: il numero di parametri aumenta con l'aumentare della dimensione del Training Set. Due esempi di modelli non-parametrici possono essere: Alberi di Decisione (Decision Tree) e Random Foresrs with the number of training set data. So far, two of the examples of non-parametric models that we have seen are: Decision Tree and Random Forest. K-Nearest Neighbours appartiene ai modelli non-parametrici che sono conosciuti come la categoria dell' **Instance-based Learning**. I modelli di **Instance-based Learning** sono capaci di registrare i dati del Training Set e sono caratterizzati da un «Lazy learning», ossia algoritmi che storano i dati e attendono finchè non ricevono un **test set**. Meno tempo in training, più tempo in predizione.

kNN (k-Nearest Neighbours)

- ✓ Possono essere usati per risolvere sia problemi di classificazione che di regressione.
- ✓ Immagazzina gli input disponibili e classifica i nuovi input sulla base di una misura di similarità come potrebbe essere una funzione distanza.
- ✓ Knn trova la sua principale applicazione nel dominio della **stima statistica** e del **pattern recognition**.
- ✓ **Passi per il kNN**:
  - ✓ KNN lavora cercando le distanze tra determinate query e tutti gli input all'interno dei dati.
  - ✓ Successivamente, seleziona uno specific numero di input, diciamo K, più vicino possible alla query richiesta.
  - ✓ Ed in seguito seleziona (vota) la label più frequente (nel caso della classificazione) o media le label (nel caso della regression).

kNN (k-Nearest Neighbours)

- ✓ Questo algoritmo di Machine Learning è particolarmente interessante dal momento che è molto differente da tutti gli altri algoritmi.

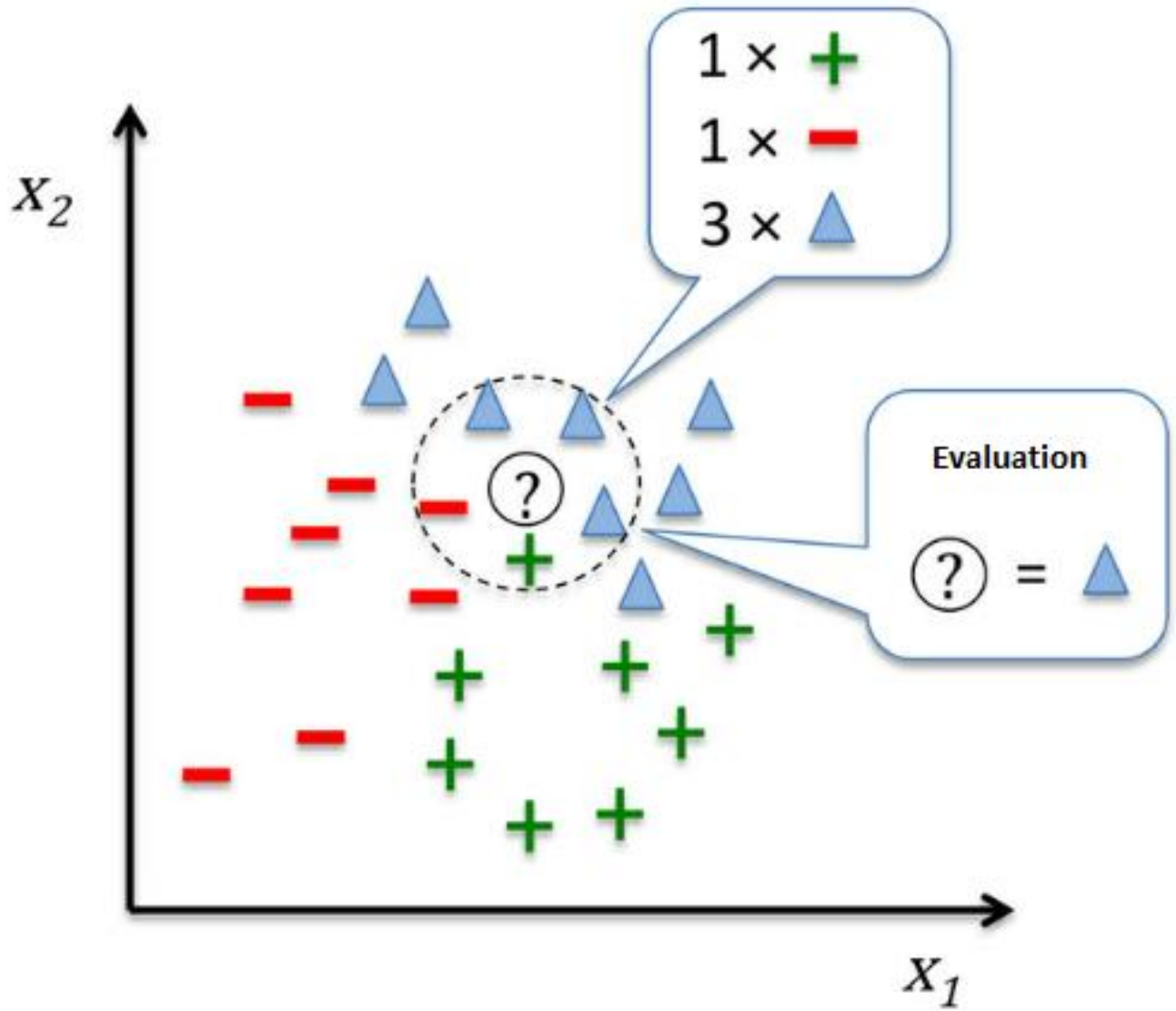
- ✓ KNN è un tipico sistema di machine learning system che rientra sotto la categoria degli algoritmi «lazy» (pigri). Questo tipo di algoritmi è denominato lazy data la sua ovvia semplicità, esso infatti non apprende una funzione di discriminazione come altri, ma «immagazzina» (stora) il data point del Training Set in qualche maniera.

- ✓ KNN può essere descritto secondo i seguenti passi algoritmici:

    1. Scegli un numero k e una metrica di distanza.
    2. Cerca i k elementi più vicini al campione che stiamo cercando di classificare.
    3. Assegna una label di classe secondo un meccanismo di elezione a maggioranza.

Nell'immagine della seguente slide vediamo il modo in cui nuovi punti di dati sono assegnati alla classe dei triangoli secondo un elezione di maggioranza tra 5 vicini (neigbours).

kNN (k-Nearest Neighbours)



Algoritmo kNN

- ✓ Carica i dati
- ✓ Inizializza k ad un numero scelto di vicini nei dati
- ✓ Per ciascun esempio nei dati, calcola la distanza tra l'esempio della query e l'input corrente proveniente dai dati
- ✓ Aggiunge quella distanza all'indice dell'input per creare una collezione ordinate di dati
- ✓ Ordina la collezione di distanze e indici in ordine crescente raggruppando per distanze
- ✓ Estrai i primi k elementi dalla collezione ordinata
- ✓ Ottieni le label delle k entry selezionate
- ✓ Se si tratta di una regressione, ritorna la **media** delle K label; se si tratta di classificazione, ritorna la **moda** delle K label

kNN (k-Nearest Neighbours)

- ✓ A seconda della metrica di distanza certa, l'algoritmo KNN cerca i k elementi del Training Set che sono più vicini (più simili) al punto che intendiamo classificare. A seconda della metrica di distanza scelta, l'algoritmo KNN trova i k esempi del training set che più si avvicinano (più simili) al data point che intendiamo classificare. La label di classe del nuovo punto è determinando per mezzo di un elezione di maggioranza tra i k elementi prossimi più vicini (k-nearest neighbours).

- ✓ Il principale vantaggio di questo approccio (basato sulla memorizzazione) è il fatto che il classificatore si adatta immediatamente e automaticamente mentre collezioniamo nuovi dati di addestramento (training data).

- ✓ Tutttavia il principale svantaggio è la complessità computazionale di questo algoritmo per classificazione di nuovi esempi, la complessità infatti cresce linearmente con l'incremento del numero di elementi nel campione all'interno del training set (nel caso peggiore), a meno che il dataset non sia costituito da un basso numero di dimensioni (features) e l'algoritmo sia implementato con strutture chiamate KD-trees.

- ✓ Inoltre, non possiamo rimuovere esempi di training, dal momento che non esiste un vero e proprio passo di addestramento. Quindi, lo spazio di memoria diviene un problema se abbiamo un enorme dataset.

kNN (k-Nearest Neighbours)

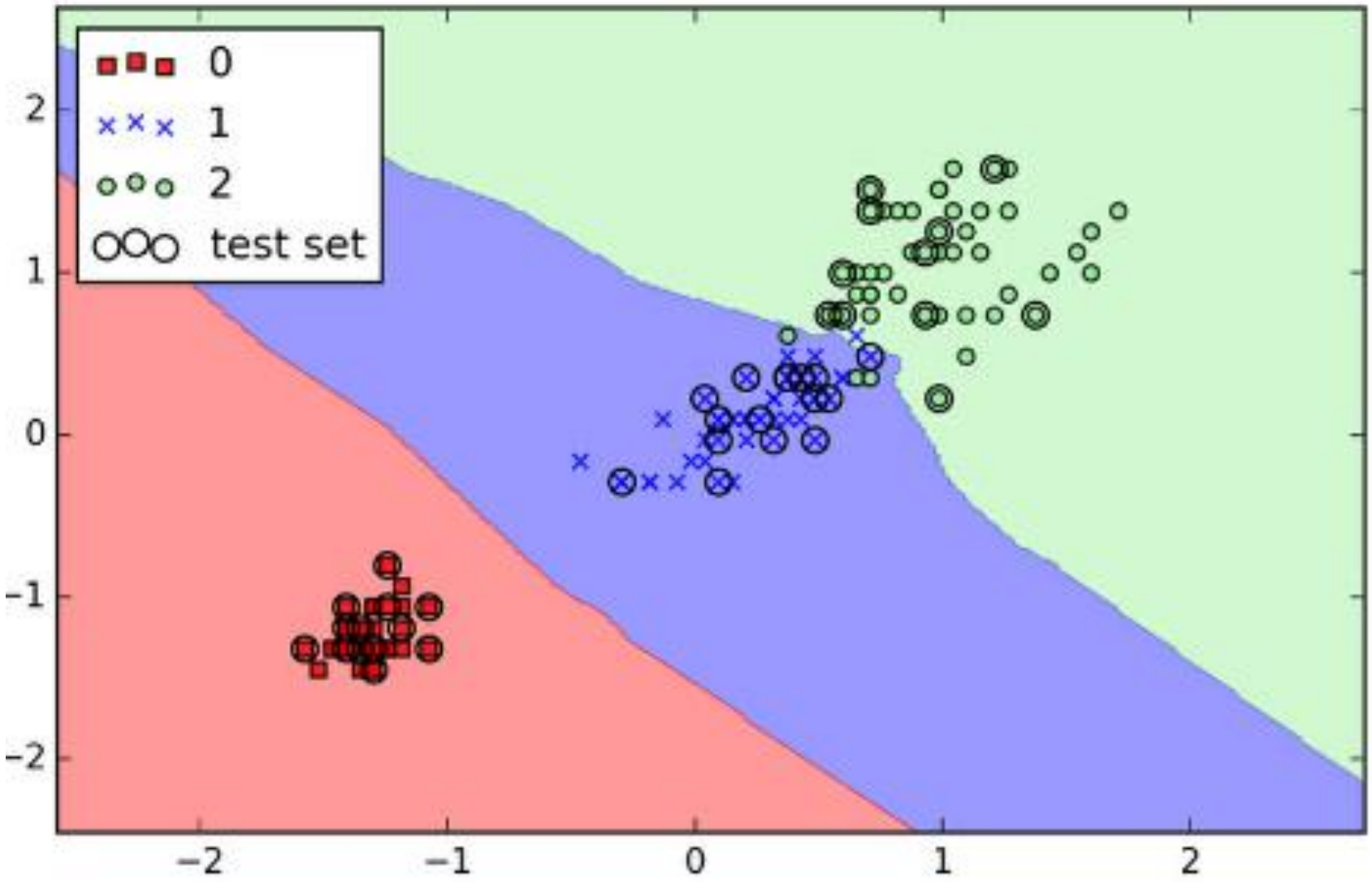
Il seguente codice implementa il modello KNN model in scikit-learn sfruttando una metrica di distanza euclidea:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_std, y_train)
plot_decision_regions(X_combined_std, y_combined, classifier=knn, test_idx=range(105,150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.show()
```

Specificando i 5 vicini più vicini nel modello KNN per il Dataset Iris, otteniamo un Decisional Boundary morbido come mostrato nella seguente figura.

kNN (k-Nearest Neighbours)



kNN (k-Nearest Neighbours)

- ✓ Nel caso di un solo nodo, l'implementazione scikit-learn dell'algoritmo KNN prediligerà i vicini che hanno una minore distanza dall'esempio della query.

- ✓ Se i vicini (neighbours) hanno una distanza simile tra loro, l'algoritmo sceglierà l'etichetta di classe che appare per prima all'interno del training set.

- ✓ La corretta scelta dei k è importante per trovare un buon bilanciamento tra i problemi di over-fitting e under-fitting.

- ✓ Inoltre, dobbiamo essere sicuri di selezionarre la giusta metrica di distanza che sia ovvero appropriata per le feature di uno specifico dataset. Spesso nei campio del mondo reale, per esempio il dataset dei fiori Iris, dove le feature sono misurate in centimetri, possiamo usare la «distanza euclidea».

- ✓ Tuttavia, se adottiamo una misura di distanza euclidea, è importante standardizzare i dati tali che ogni feature influenzi uniformemente la distanza. La **distanza di Minkowski** che abbiamo usato nel precedente codice è semplicemente una generalizzazione della distanza Euclidea (o distanza di Manhattan).

$$d\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}\right) = \sqrt[p]{\sum_k \left|x_k^{(i)} - x_k^{(j)}\right|^p}$$

kNN (k-Nearest Neighbours)

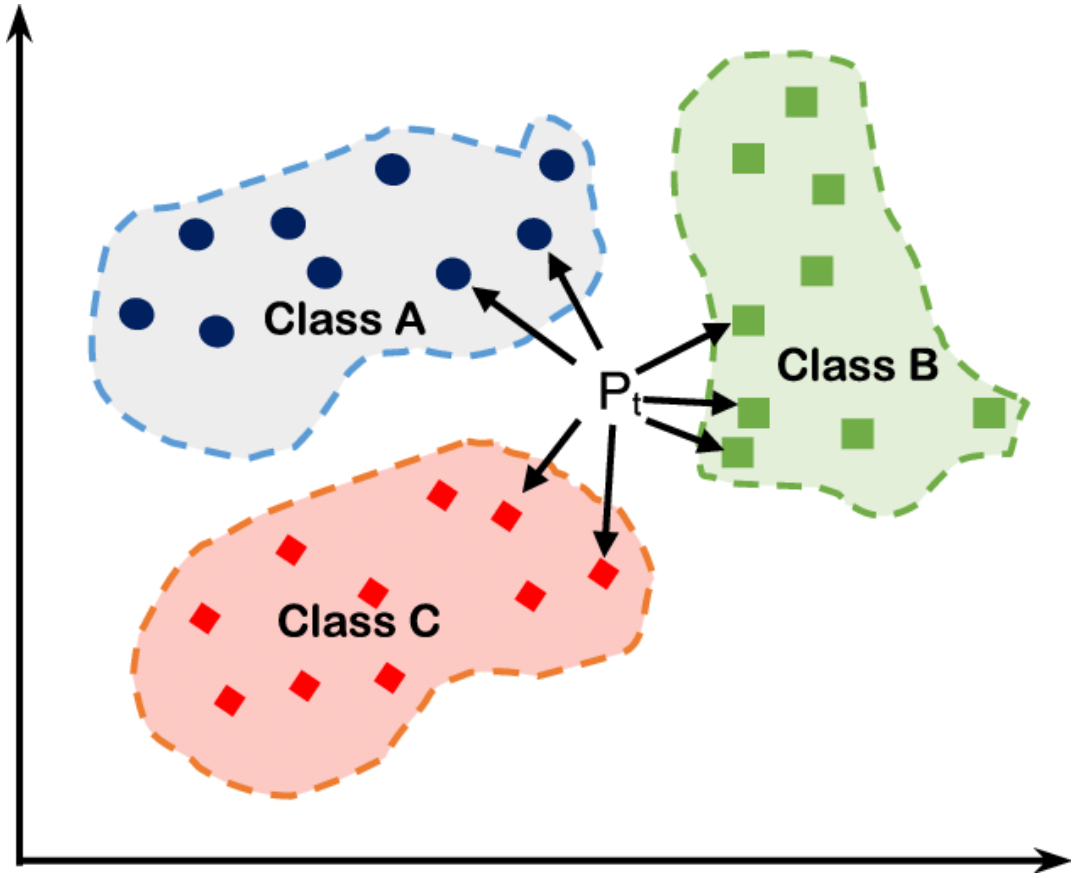
- ✓ La distanza di Monkwski diventa una distanza euclidea quando il parametro p=2 o la distanza mancante (missing distance) quando p=1.

- ✓ Molte altre valutazioni di distanze sono disponibili su scikit-learn e possono essere usate con un parametro della metrica. Sono elencate di seguito: http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

- ✓ E' importante menzionare che KNN è soggetto al problema dell'overtting problem, in caso di alta dimensionalità. E' purtroppo un effetto dovuto alle situazioni in cui lo spazio delle feature diviene sempre più sparso a causa della dimensione di un training set fissato. Intuitivamente, possiamo considerare che i «nearest neighbours» sono troppo lontani in uno spazio di grandi dimensioni, e questo non porta ad ottenere buone stime.

- ✓ La regolarizzazione è una tecnica per la regressione logistica che porta a ridurre l'over-fittin, tuttavia in modelli senza metodi di regolarizzazione disponibili come per gli Alberi di Decisione o i KNN, possiamo usare tecniche per selezionare alcune feature e ridurre la dimensionalità (PCA) al fine di mitigare il problema delle elevate dimensionalità.

kNN



Applicazioni of kNN

- ✓ Individuazione di Impronte
- ✓ Predizioni di Azioni Finanziarie
- ✓ Predizione di Tasso di Campio della Moneta
- ✓ Predizione di Bancarotte Bancarie
- ✓ Gestione del Credito e dei Prestiti
- ✓ Analisi di Anti-Riciclaggio del Denaro
- ✓ Stima della quantità di glucosio nel sangue di una persona diabetica a partire dallo spettro di assorbimento IR del sangue di una persona.
- ✓ Identificare i fattori di rischio per un cancro basato su variabili cliniche e demografiche.

Bibliografia

- ✓ https://towardsdatascience.com/top-10-algorithms-for-machine-learning-beginners-149374935f3c
- ✓ J. H. Friedman, J. L. Bentley e R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. "ACM Transactions on Mathematical Software (TOMS)", 3(3):209–226, 1977

Francesco **Pugliese**