SUPSI

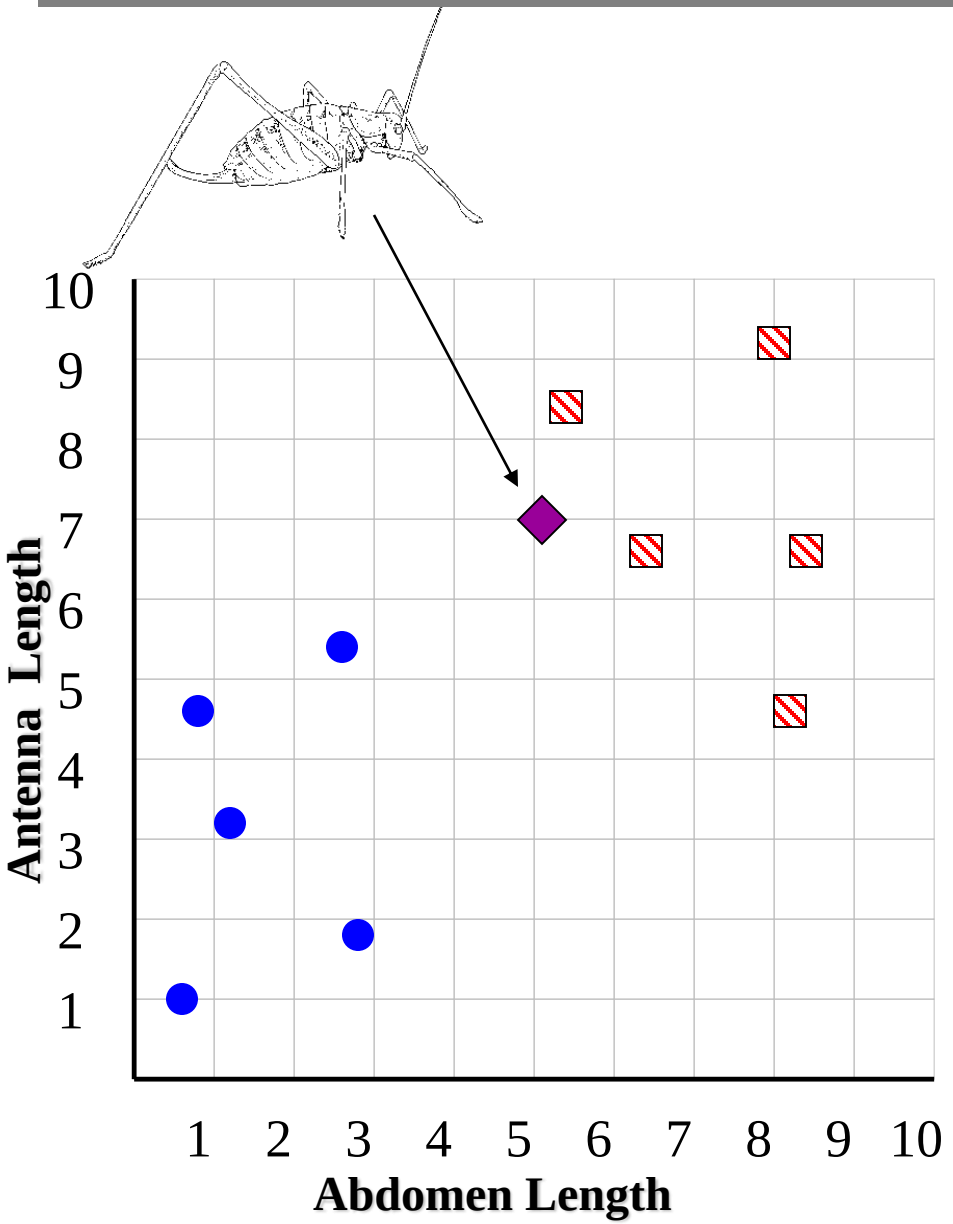
Data Science

Metodi di classificazione: Il classificatore Nearest Neighbor

Alessandro Giusti
IDSIA – SUPSI, Galleria 1, Manno
alessandro.giusti@supsi.ch

Parte del materiale tratto da:
F. Stella, "Business Intelligence", corso di Laurea in Informatica, Università degli studi di Milano - Bicocca
L. Azzimonti. Slide per il corso di "Business Intelligence". Corso di Laurea in Ing. Gestionale. SUPSI
E. Keogh. Introduction to Machine Learning.
G. Corani

previously unseen instance =	11	5.1	7.0	???????



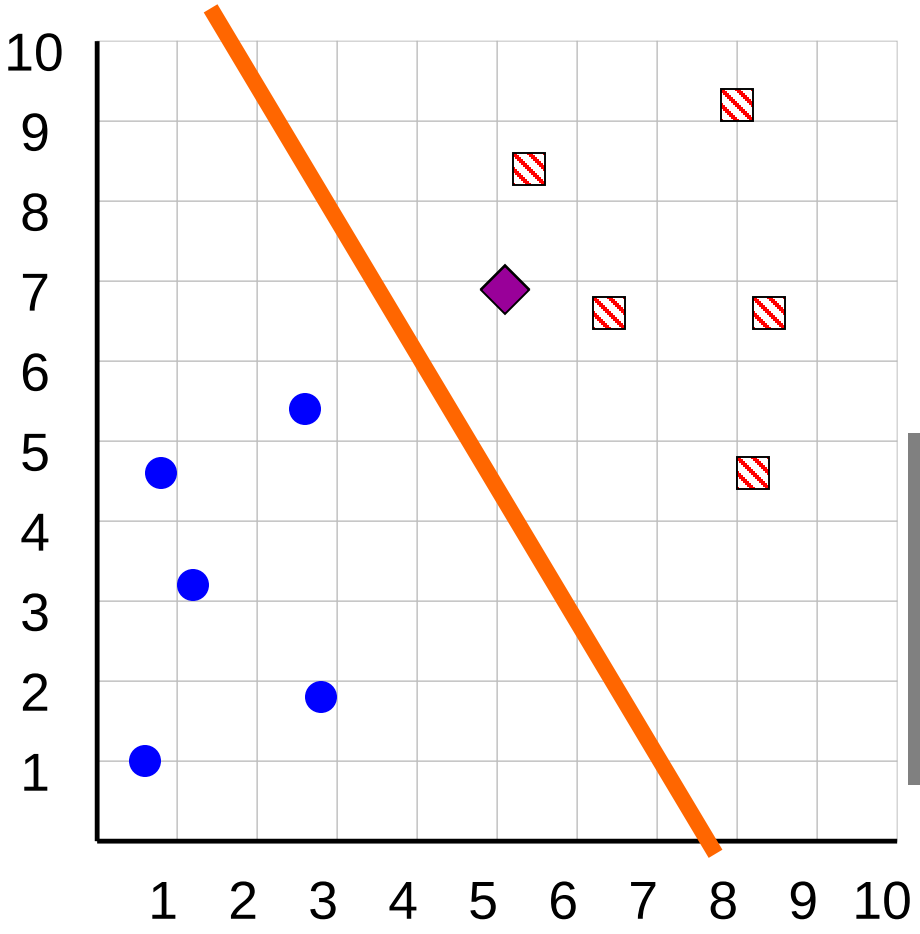
- Possiamo "proiettare" l'osservazione che non abbiamo ancora visto nello stesso spazio del dataset

- Ora possiamo dimenticare i dettagli specifici del problema, e ragionare solo con i punti nello spazio!

▨ **Katydids**

● **Grasshoppers**

Classificatore lineare semplice

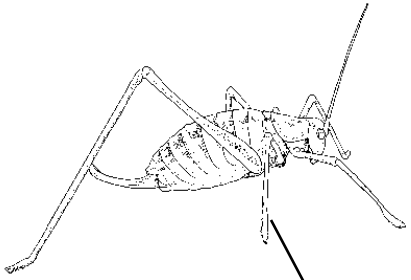


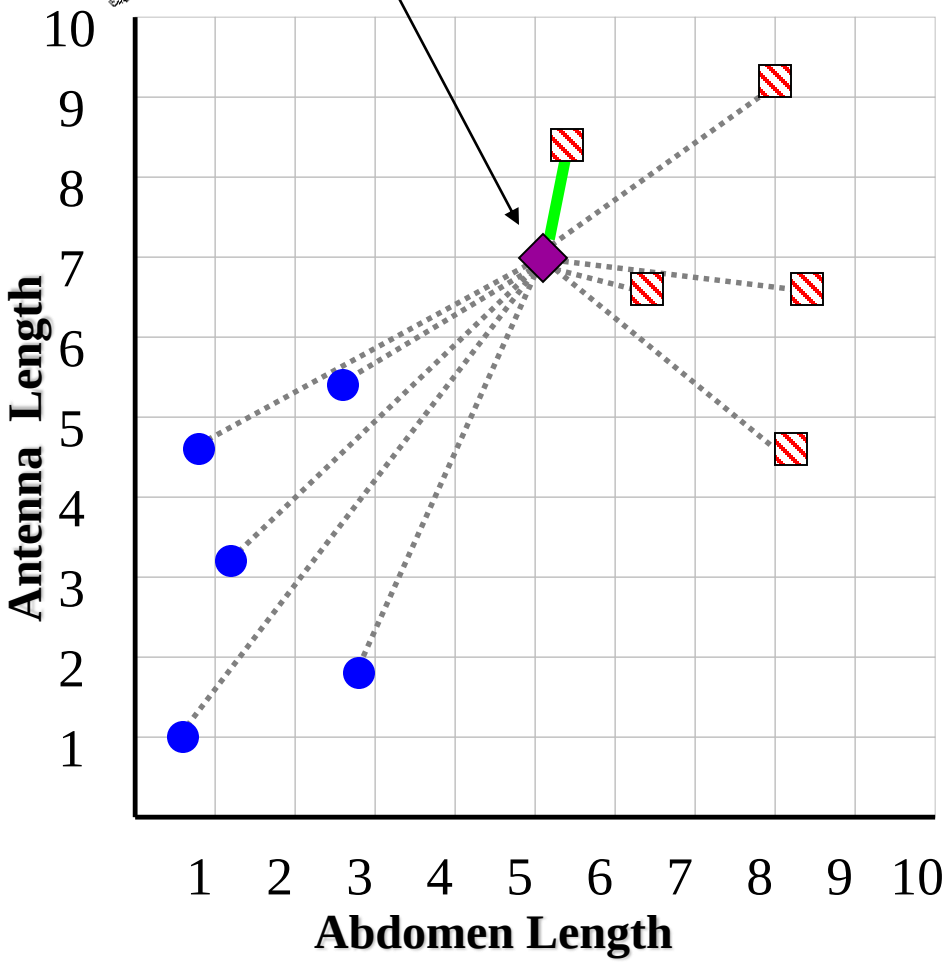
R.A. Fisher
1890-1962

If nuova osservazione sopra la linea
then
class is **Katydid**
else
class is **Grasshopper**

Katydids
Grasshoppers

Nearest Neighbor Classifier





Evelyn Fix
1904-1965

Joe Hodges
1922-2000

Se l'osservazione piu' vicina a quella di test
e' un Katydid
classifico come **Katydid**
altrimenti
classifico come **Grasshopper**

Katydids
Grasshoppers

Esempio

Istanze di training				
ATTRIB1	**ATTRIB 2**	**ATTRIB 3**	**ATTRIB 4**	**CLASS**
0.4	0.5	0.6	0.3	C_1
0.1	0.6	0.9	0.3	C_0
0.1	0.1	1.0	0.8	C_1

Istanza di test (Query)				
ATTRIB1	**ATTRIB 2**	**ATTRIB 3**	**ATTRIB 4**	**CLASS**
0.6	0.4	0.0	0.3	?

- Calcola la distanza euclidea tra ciascuna osservazione di training e l'osservazione di test. La distanza e' calcolata solo nello spazio degli attributi esplicativi (features), non dipende dalla classe!
- Ad esempio, la distanza euclidea tra l'osservazione di test e la prima istanza di training e':

$$\sqrt{(0.6-0.4)^2+(0.5-0.4)^2+(0.6)^2+0^2}$$

Esempio

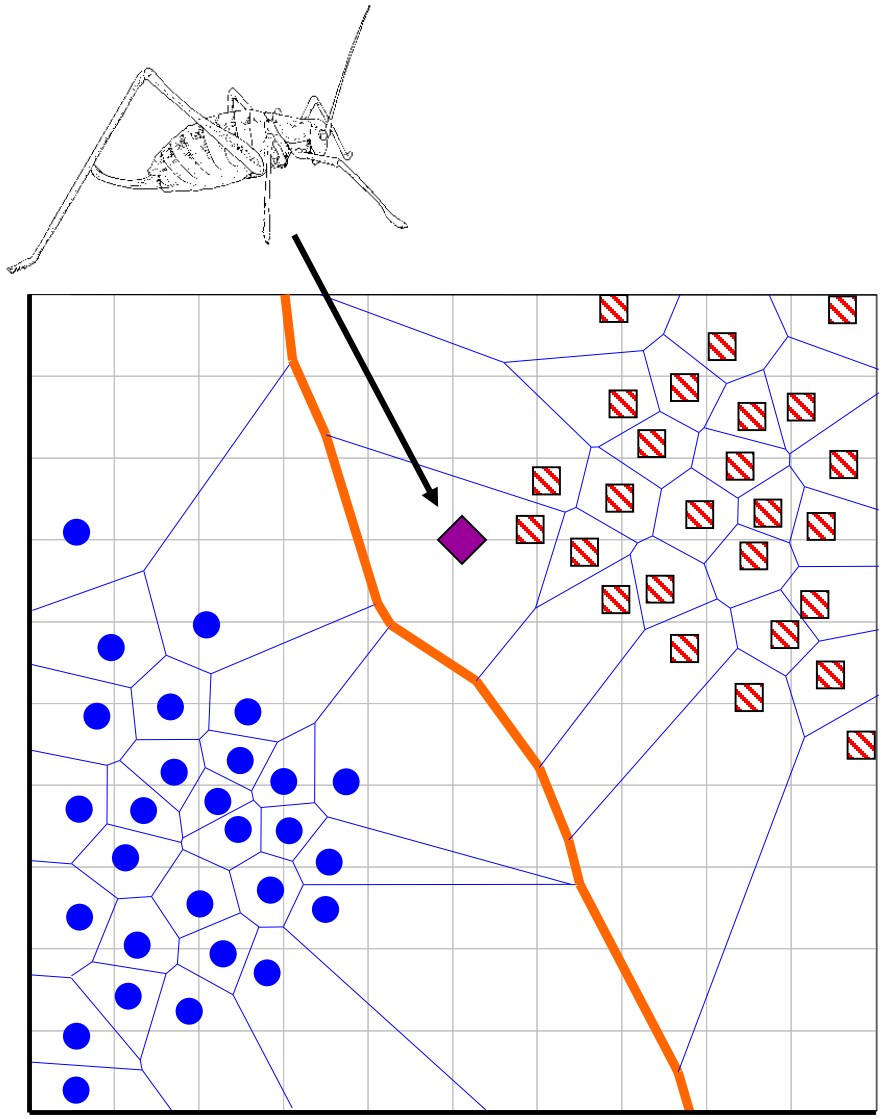
Istanze di training				
ATTRIB1	**ATTRIB 2**	**ATTRIB 3**	**ATTRIB 4**	**CLASS**
0.4	0.5	0.6	0.3	C_1
0.1	0.6	0.9	0.3	C_0
0.1	0.1	1.0	0.8	C_1

Istanza di test (Query)				
ATTRIB1	**ATTRIB 2**	**ATTRIB 3**	**ATTRIB 4**	**CLASS**
0.6	0.4	0.0	0.3	?

- Le tre istanze di training hanno queste distanze da quella di test, rispettivamente: 0.64, 1.04, 1.26.

- Scegli la piu' vicina (la prima), e classifica con la classe corrispondente (C_1).

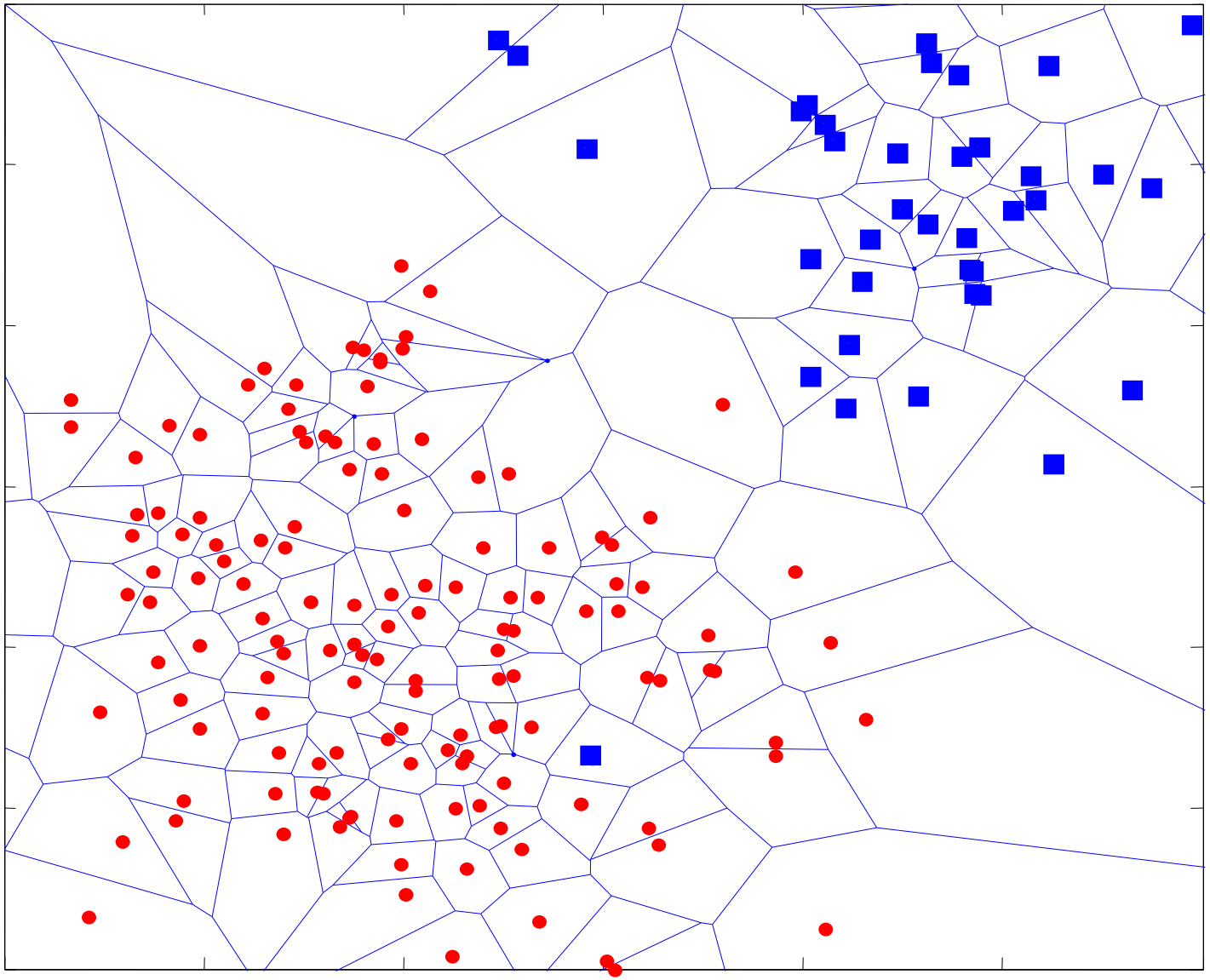
He forma ha il decision boundary?



Nota che non dobbiamo esplicitamente costruire questo bordo! Emerge dalla regola di cui abbiamo parlato prima

Questa partizione dello spazio si chiama diagramma di voronoi

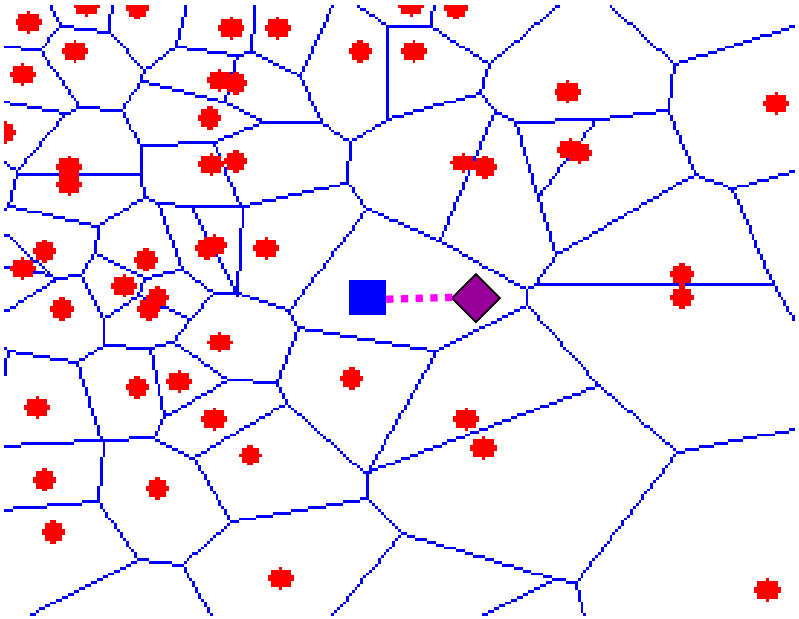
Ma occhio: questo algoritmo non e' robusto agli outlier!



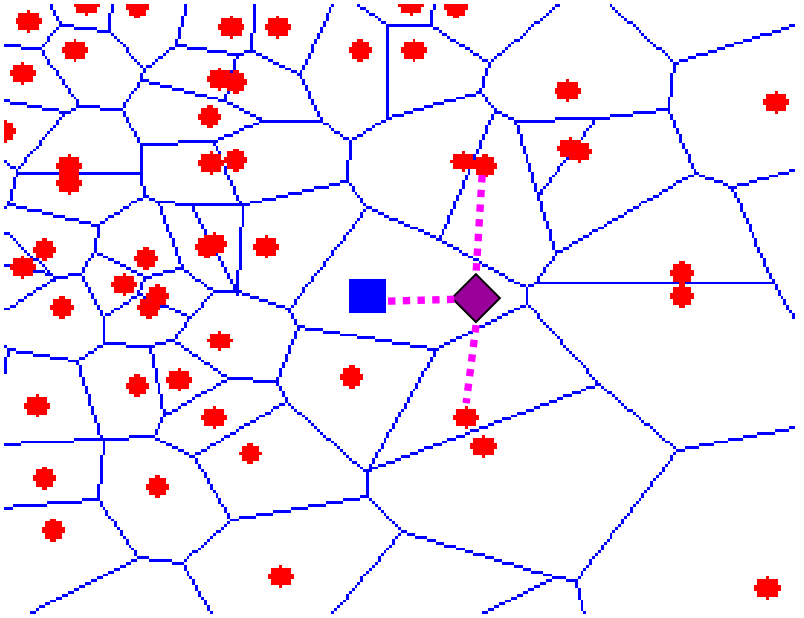
La soluzione e'…

Generalizziamo l'algoritmo Nearest Neighbor: K-nearest neighbor (KNN) algorithm.

Misuriamo la distanza alle K osservazioni piu' vicine, e le lasciamo votare. K di solito e' dispari



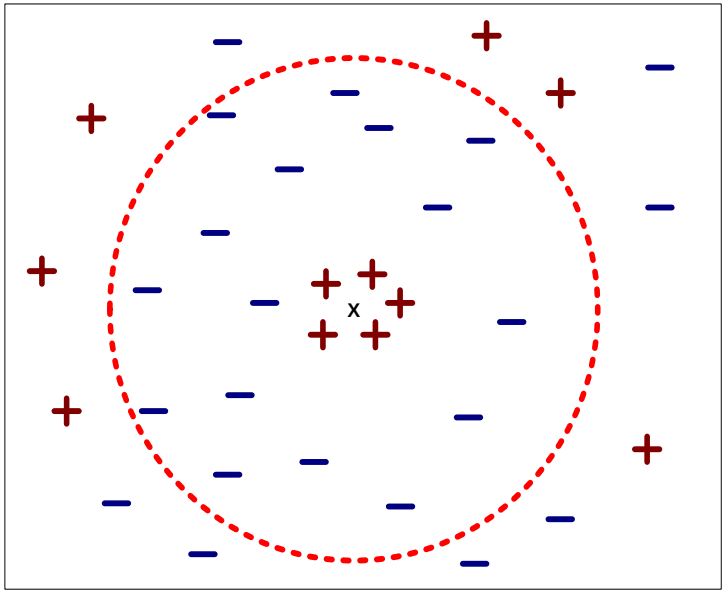
K = 1



K = 3

Ma come scegliamo k?

- Un k piccolo basa la scelta su poche istanze. Ciononostante, 1-NN e' usato molto spesso.
- Per k troppo grande c'e' il rischio di considerare neighbor molto lontani che non dovremmo tenere in considerazione
- Scelte tipiche: k=1, 5, 10.

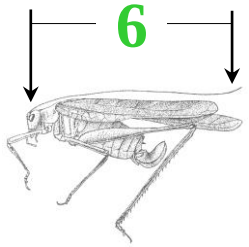


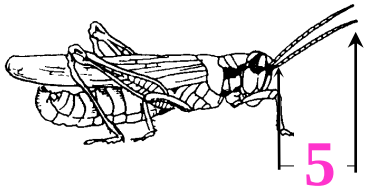
L’algoritmo nearest neighbor non e’ robusto alle feature irrilevanti

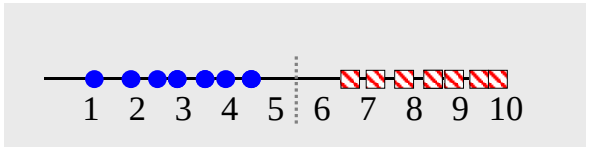
Ipotizziamo che valga la regola: se la lunghezza delle antenne e’ piu’ di 5.5 abbiamo sempre **Katydid**, altrimenti **Grasshopper**.

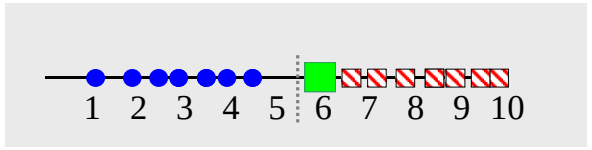
Possiamo ottenere un classificatore perfetto usando solo questa feature!

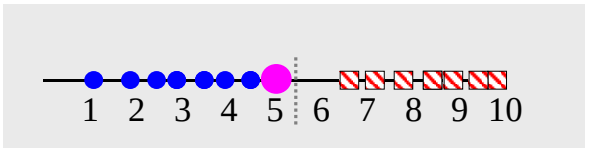
Training data

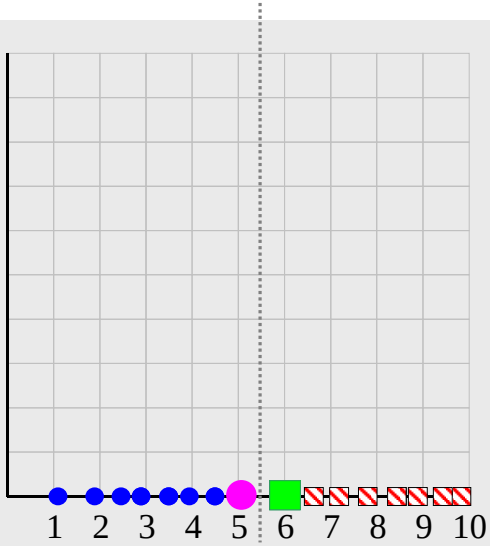


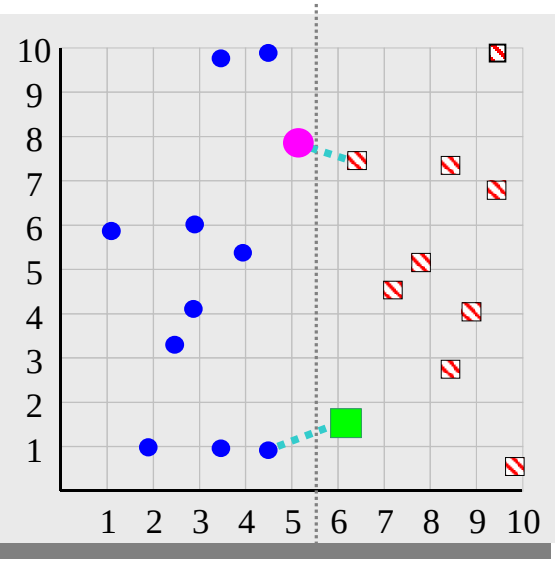












Pero’ aggiungiamo una feature **irrilevante**, come ad esempio il peso dell’insetto

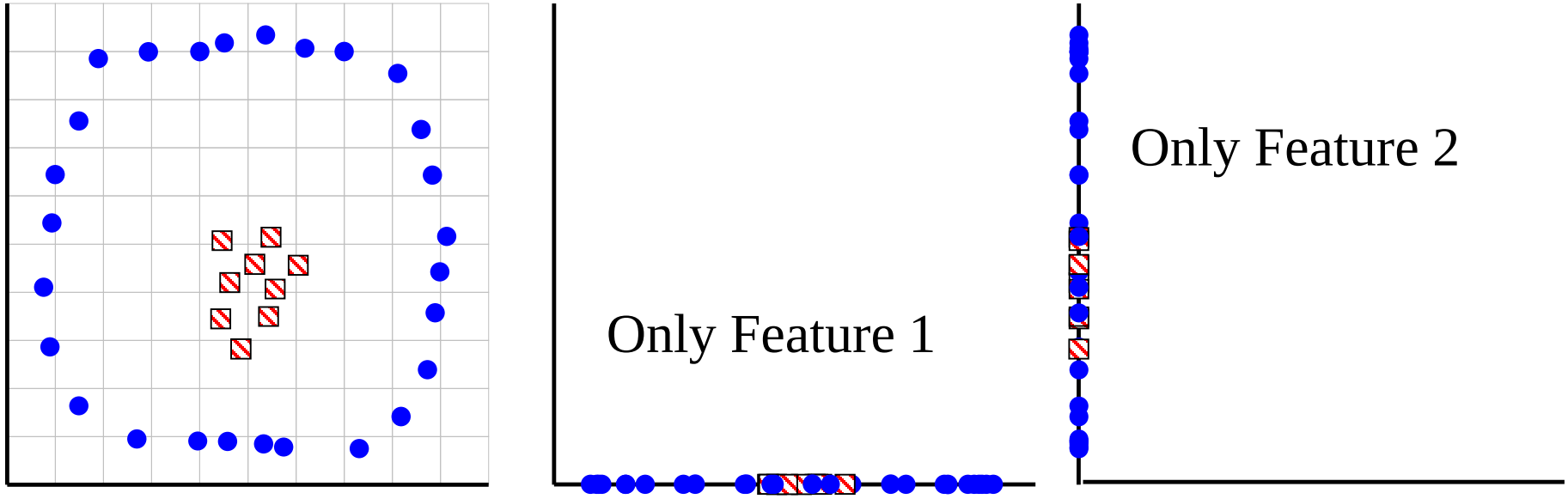
Ma adesso usando entrambe le feature otteniamo un classificatore imperfetto!

Come risolvere questo problema?

- Usiamo piu' osservazioni per addestrare il classificatore
- Chiediamo a un esperto di dirci quali feature sono potenzialmente importanti
- Troviamo automaticamente le feature importanti nei dataset (*feature selection*)
- Proviamo a caso alcuni sottoinsiemi tra le feature disponibili (ma nella prossima slide scopriamo che non e' facile)

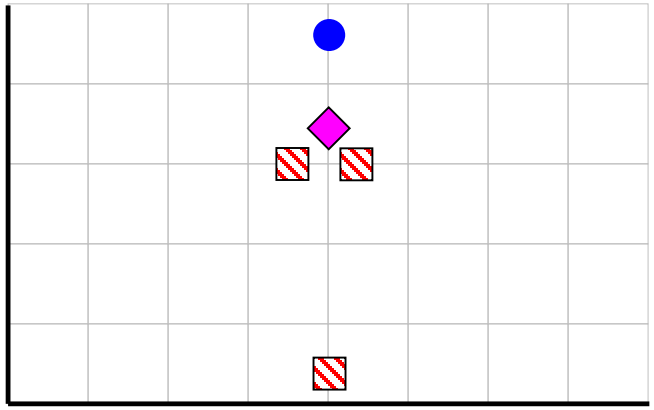
Perche' e' difficile cercare i sottoinsiemi di feature?

Abbiamo un problema con 100 features, dove le features 1 e 2 (X e Y qui sotto) permettono una classificazione perfetta, ma tutte le altre 98 sono irrilevanti.



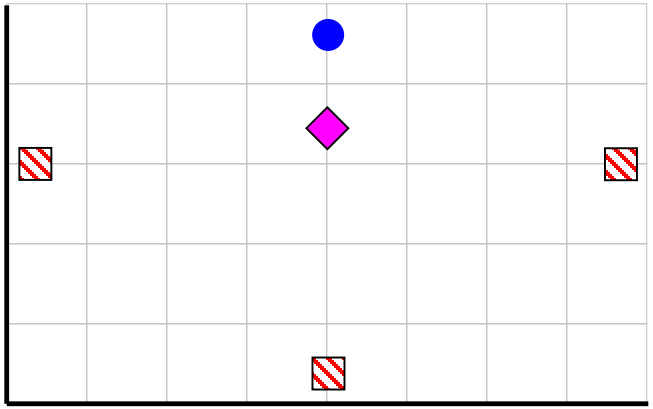
Usare tutte e 100 le features funziona male, cosi' come usare solo la feature 1 o la feature 2 o qualsiasi delle altre singolarmente. Solo uno dei 2^{100} –1 possibili sottoinsiemi funziona bene!

L’algoritmo nearest neighbor e’ sensibile alle unita’ di misura!



Asse X in **centimetri**
Asse Y in dollari

Il nearest neighbor all’osservazione **rosa** e’ quella **rossa**.



Asse X in **millimetri**
Asse Y in dollari

Il nearest neighbor all’osservazione **rosa** e’ quella **blu**.

Il problema si risolve *standardizzando* i valori

Standardizzazione

- *Standardizzare* i valori ci permette di non essere dipendenti dalle unita' di misura e di confrontare attributi espressi con unita' di misura diverse
  - Esempio:
    - "altezza" di una persona puo' variare da 1.5m a 1.8m
    - "peso" da 90lb a 300lb
    - "reddito" da $10K a $1M

- Dopo la standardizzazione ogni attributo ha media 0 e deviazione standard 1

- Gli attributi diventano numericamente omogenei e confrontabili

Standardizzazione

- Per standardizzare: sottraggo la media e divido per la dev. Std.
- Cosa succede alle unita' di misura? Ottengo un valore adimensionale!
- Dopo la standardizzazione ho media 0 e dev. Std 1

$$z_i = \frac{x_i - \bar{x}}{\sigma_x} \qquad \text{where:}$$

z_i and x_i are the value of the attribute on the i-th instance, after and before standardizing

$\bar{x}$ is the mean of the attribute on the training set

σ_x is the std dev of the attribute on the training set.

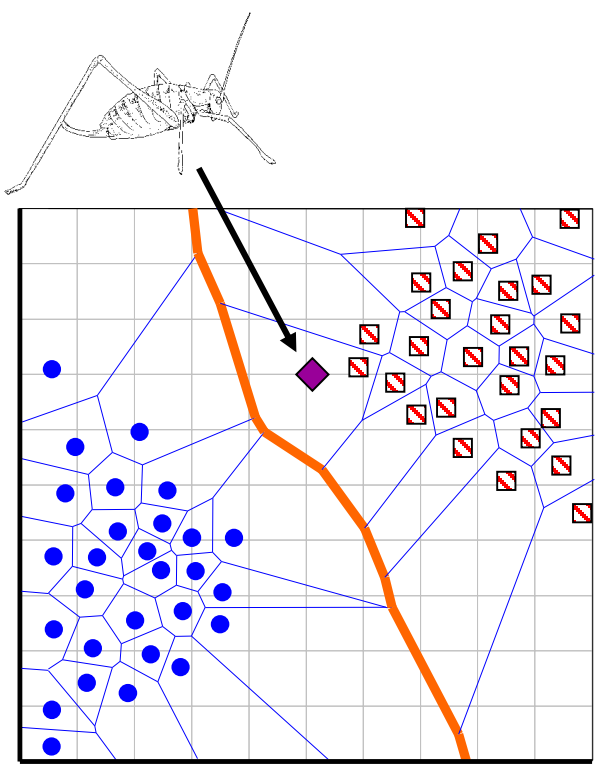
Standardizzazione: ricetta

1. Standardizza ogni attributo del training set → Training set standardizzato
2. Memorizza media e deviazione standard che hai usato per standardizzare ogni feature
3. Prima di classificare un'istanza di test, standardizza ciascun attributo usando la media e deviazione standard calcolata precedentemente
4. A questo punto puoi calcolare la distanza tra la query standardizzata e ciascun istanza del training set standardizzato

Nota: la standardizzazione e' utile per molti altri algoritmi, non solo nearest neighbor!

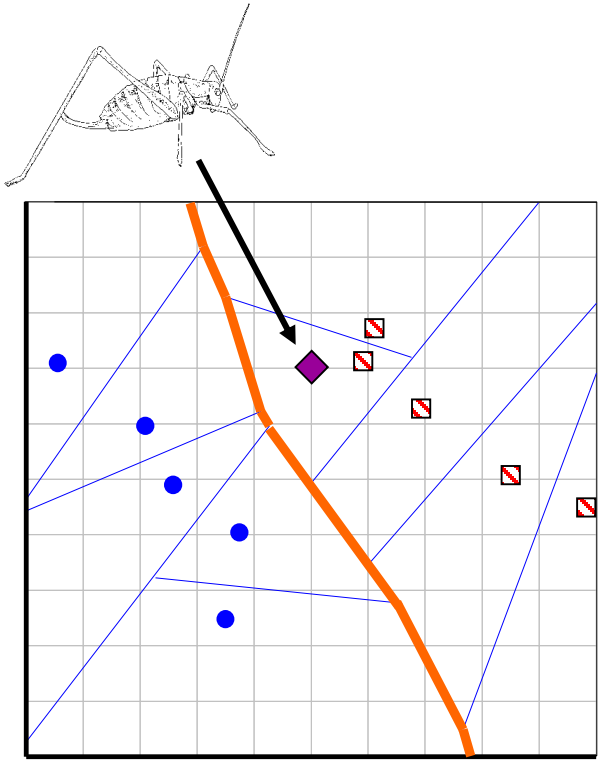
Torniamo all'algoritmo Nearest Neighbor.

Possiamo accelerare l'algoritmo buttando via dei dati: talvolta puo' persino aumentare l'accuratezza



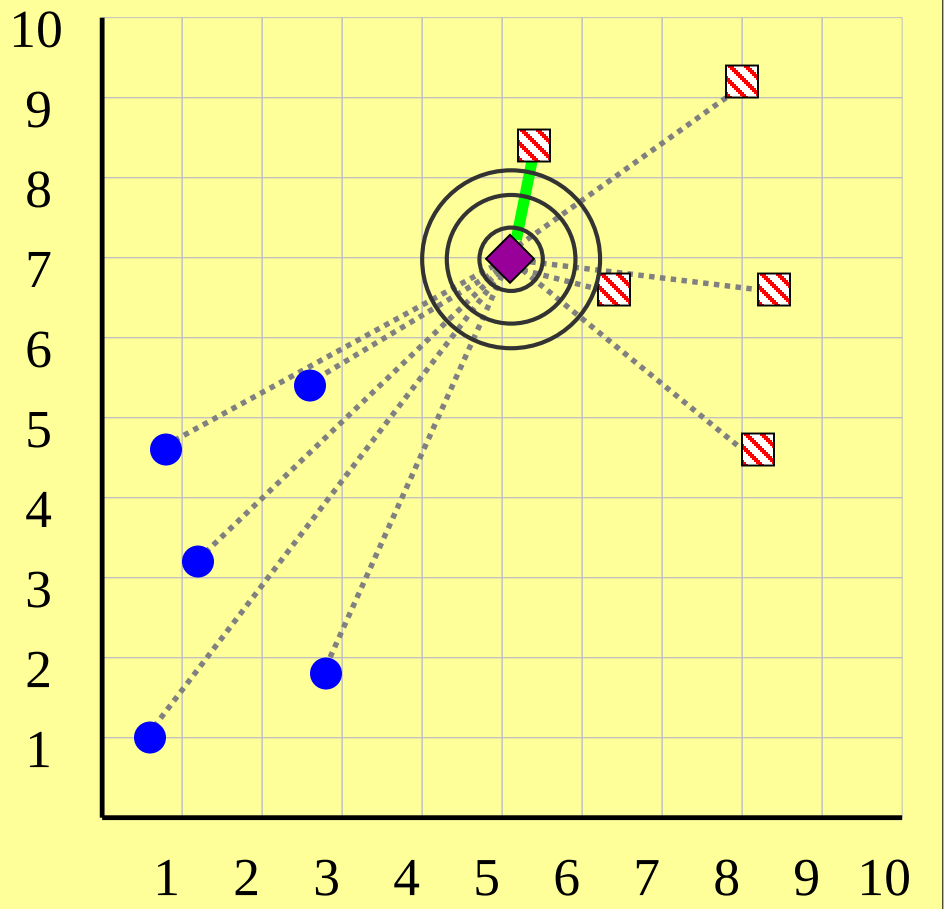
Un possibile approccio: Dimentica tutte le osservazioni circondate da osservazioni della stessa classe

Nota che nell'esempio, il decision boundary cambia (si semplifica)



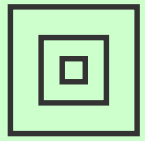
Finora abbiamo usato la distanza euclidea, ma non e' l'unica scelta

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$

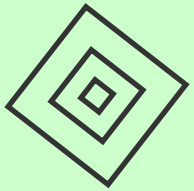


$$D(Q,C) \equiv \sqrt[p]{\sum_{i=1}^{n}(q_i - c_i)^p}$$

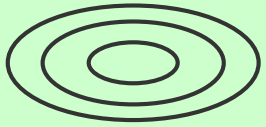
Max (p=inf)



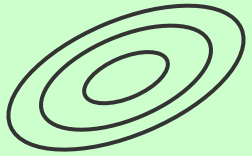
Manhattan (p=1)



Weighted Euclidean



Mahalanobis



… possiamo usare l'algoritmo nearest neighbor con qualsiasi misura di distanza!

"*Faloutsos*" e' greco or irlandese? Potremmo confrontare "*Faloutsos*" con un database di nomi utilizzando come misura la *edit distance*…

edit_distance(*Faloutsos*, *Keogh*) = 8
edit_distance(*Faloutsos*, *Gunopulos*) = 6

Possiamo ipotizzare che la somiglianza con altri nomi Greci indichi che *Faloutsos* e' a sua volta greco

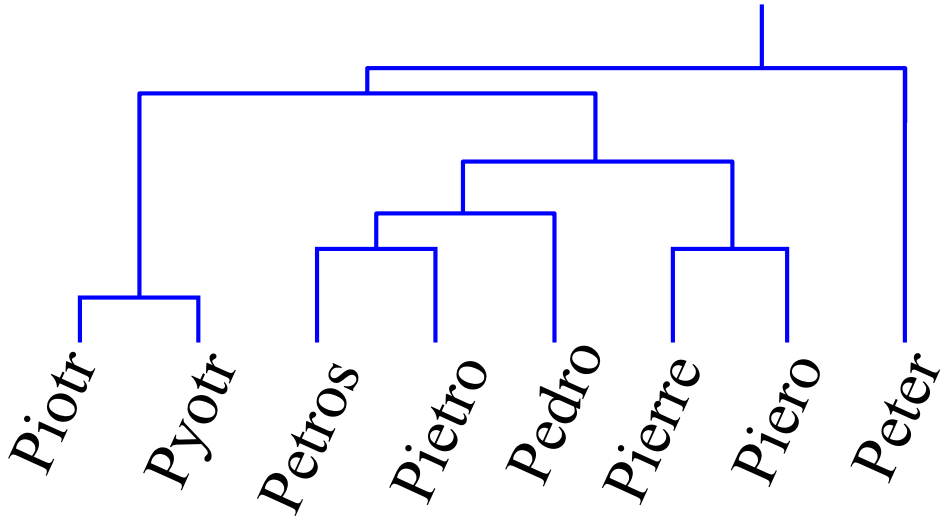
ID	Name	Class
1	Gunopul**os**	**Greek**
2	Papadopoul**os**	**Greek**
3	Kolli**os**	**Greek**
4	Dardan**os**	**Greek**
5	Keo**gh**	**Irish**
6	Gou**gh**	**Irish**
7	Greenhau**gh**	**Irish**
8	Hadlei**gh**	**Irish**

Esistono misure di distanza specializzate per stringhe, serie temporali, immagini, grafici, video, insiemi, impronte digitali, sequenze DNA

Esempio: edit distance

Possiamo trasformare ogni stringa *Q* nella stringa *C*, usando solo *Substitution*, *Insertion* e *Deletion*.
Assegnamo un costo a ciascuna di queste operazioni

La somiglianza tra le due stringhe e' data dal costo della trasformazione piu' economica tra le due (nota: non trattiamo il modo di trovare questa trasformazione)

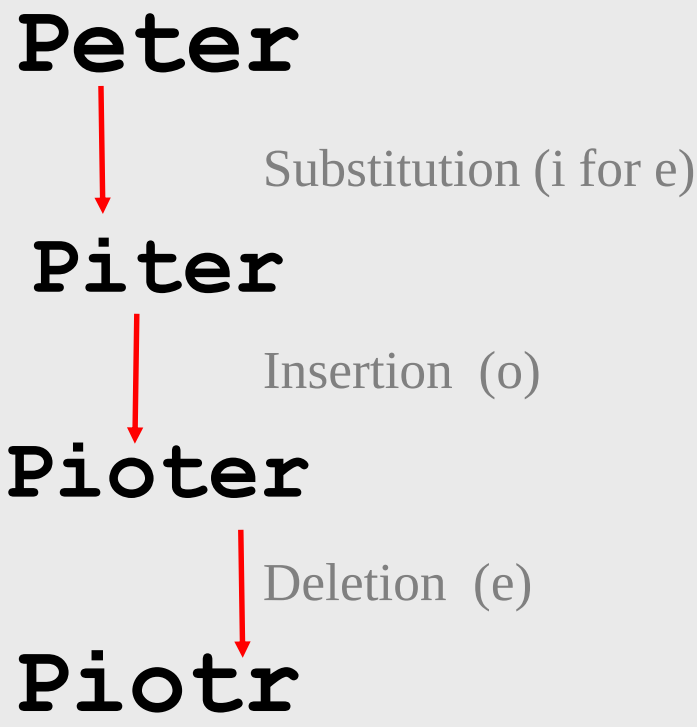


Quanto sono simili "Peter" e "Piotr"?

Ipotizza I seguenti costi

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(**`Peter,Piotr`**) e' 3



Weighted k-nearest neighbors

- Per classificare un'istanza, assegno un peso diverso a ciascuno dei k nearest neighbor, proporzionalmente all'inverso della distanza
- Predico la classe per cui la somma dei pesi e' massima.

Neighbor	Class	Distance from query	Inverse of distance (weight)
1	Yes	0.1	10
2	No	0.2	5
3	Yes	0.5	2

- Peso di Yes : 10 +2 =12
- Peso di No : 5
- Predizione: Yes.

Weighted k-nearest neighbors

- In alternativa posso usare l'inverso della distanza al quadrato (non c'e' consenso su quale sia la soluzione migliore)

Neighbor	Class	Distance from query	Squared distance	Inverse of squared distance
1	Yes	0.1	0.01	100
2	No	0.2	0.04	25
3	Yes	0.5	0.25	4

- Peso di Yes : 100 +4 =104
- Peso di No : 25
- Predizione: Yes.

Note e considerazioni

- k-NN e' un *lazy learner*: non fa nulla finche' non e' richiesto di classificare un'osservazione di test

- Requisiti di memoria; il training set va tenuto tutto in memoria (a meno di non scegliere di dimenticare delle osservazioni, come visto in precedenza), che puo' essere problematico