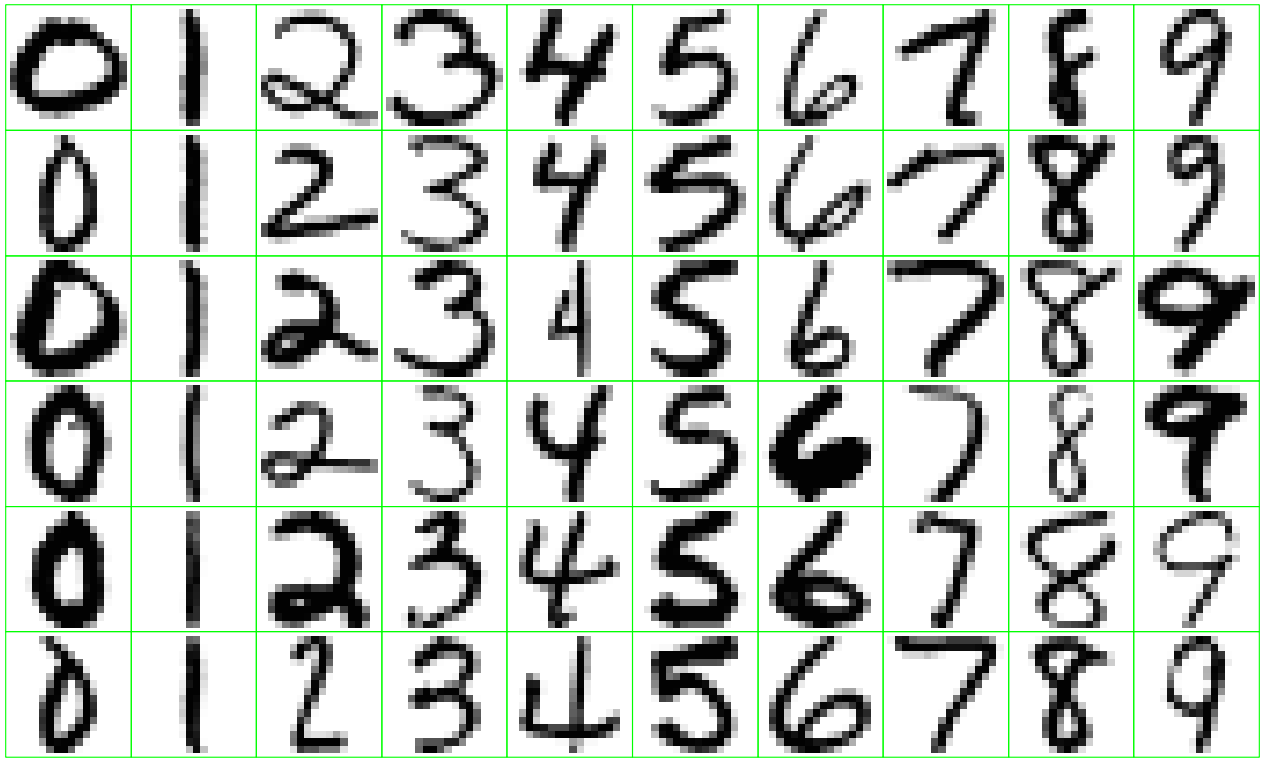
Introduzione al Machine Learning: Classificazione Nearest Neighbor

Vincenzo Bonifaci

Esempio: Riconoscimento di cifre scritte a mano

Input: immagine 28×28 in scala di grigi
Output: la cifra decimale (0–9) rappresentata dall'immagine



Dataset MNIST: 60,000 (training set) + 10,000 (test set) immagini etichettate

Problemi di predizione: input e output

- Spazio degli input $\mathcal{X}$
  Es.: insieme delle possibili immagini 28×28
- Spazio degli output $\mathcal{Y}$
  Es.: $\{0, 1, 2, \ldots, 9\}$

Dopo aver visto un certo numero di esempi (x, y), vogliamo trovare una *regola di predizione* (o *ipotesi*)

$$h : \mathcal{X} \to \mathcal{Y}$$

che ricostruisca in maniera accurata la relazione ingresso-uscita

Nei problemi di *regressione* l'output è quantitativo

Nei problemi di *classificazione* l'output è qualitativo

Funzioni di costo [loss functions]

Come quantifichiamo l'accuratezza di una regola di predizione $h : \mathcal{X} \to \mathcal{Y}$ su un particolare esempio?

Una *funzione di costo* è una funzione ℓ che prende una regola di predizione h ed un esempio $(x, y) \in \mathcal{X} \times \mathcal{Y}$, e restituisce un reale nonnegativo

$$\ell(h, (x, y)) \in \mathbb{R}_+$$

Una funzione di costo per la classificazione

- *Funzione costo 0-1*:

$$\ell(h,(x,y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{se } h(x) = y \\ 1 & \text{se } h(x) \neq y \end{cases}$$

Il *rischio empirico* diventa la frazione di esempi di training non correttamente classificati:

$$\mathrm{RE}_S(h) = \frac{|\{(x,y) \in S : h(x) \neq y\}|}{|S|}$$

Il *rischio atteso* diventa la probabilità che un nuovo esempio non sia correttamente classificato (*inaccuratezza* del classificatore):

$$\mathrm{RA}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

Classificazione Nearest-Neighbor

Classificazione Nearest Neighbor

Immagini di training $x^{(1)}, x^{(2)}, \ldots, x^{(60000)}$
Etichette $y^{(1)}, y^{(2)}, \ldots, y^{(60000)}$ (numeri nel range 0–9)

Come classifichiamo una nuova immagine x?
Approccio Nearest Neighbor:

- Trova l'esempio più "simile" ad x tra gli $x^{(i)}$
- Restituisci la corrispondente etichetta

Lo spazio dei dati

Come misuriamo la distanza tra immagini?

- Dimensioni 28×28 (784 pixel totali)
- Ogni pixel è in scala di grigi: 0–255

Un vettore 784-dimensionale per ogni immagine

- Spazio degli input $\mathcal{X} = \mathbb{R}^{784}$
- Spazio degli output (etichette) $\mathcal{Y} = \{0, 1, \ldots, 9\}$

La distanza euclidea tra x e x' è $\|x - x'\| = \sqrt{\sum_k (x_k - x'_k)^2}$

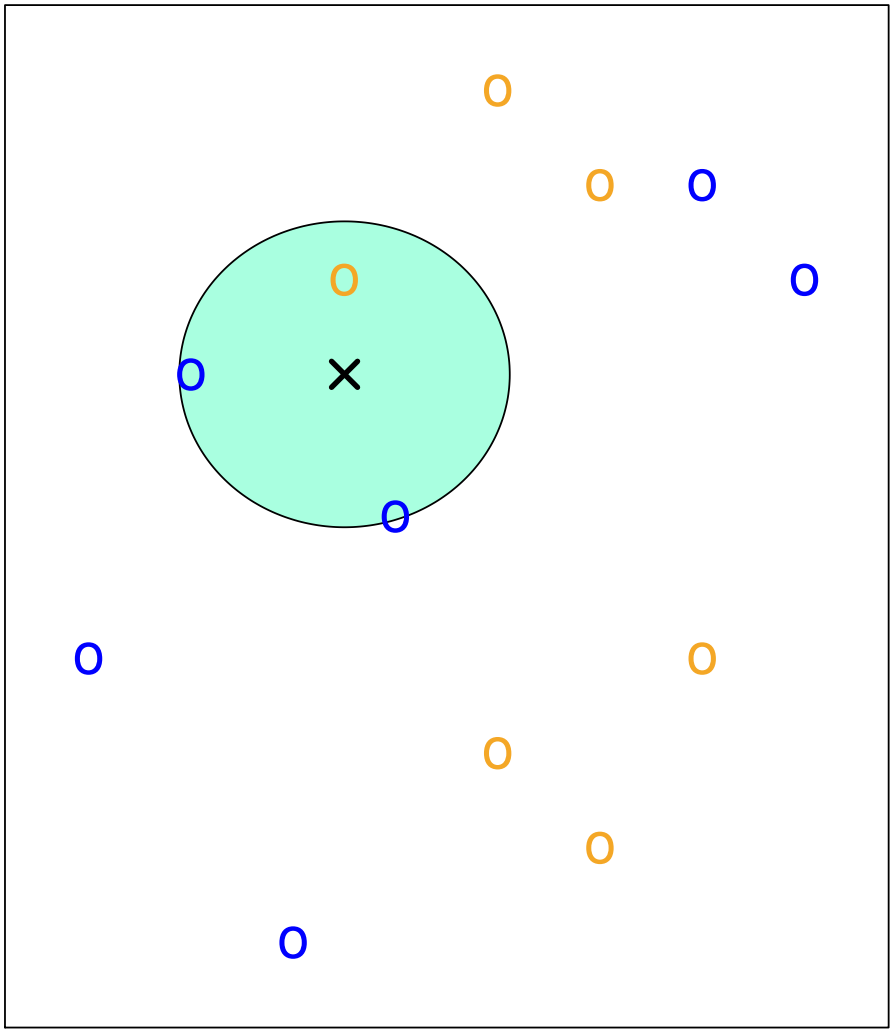
Classificazione K-Nearest Neighbor (K-NN)

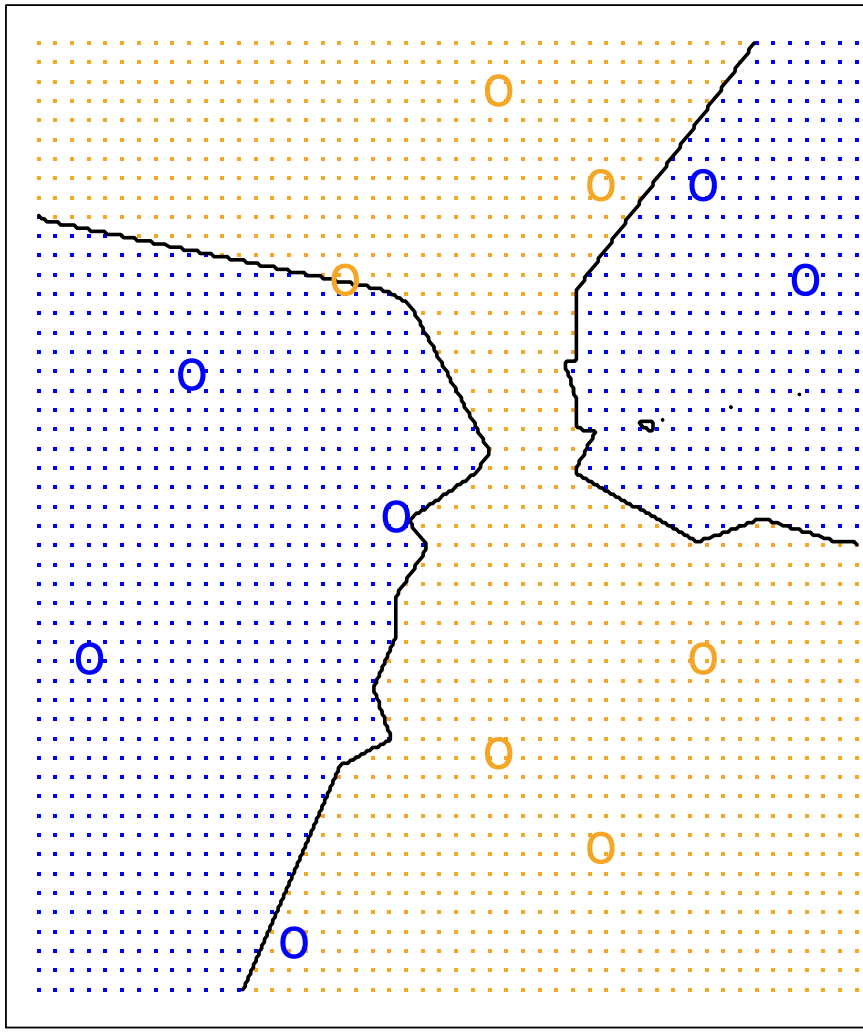
Sia $K \geq 1$ e sia x il punto di cui si vuole stimare l'etichetta

1. Identifica i K esempi $x^{(1)}, \ldots, x^{(K)}$ più vicini ad x (in termini di distanza euclidea)
2. Restituisci l'etichetta più frequente per quegli esempi: $h(x) = \text{argmax}_{y \in \mathcal{Y}} |\{i = 1, \ldots, K \ : \ y^{(i)} = y\}|$

Quando $|\mathcal{Y}| = 2$, l'ultimo passo equivale a restituire l'etichetta di maggioranza

K-NN: Esempio ($K = 3$)





Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Che rischio atteso avrebbe un classificatore totalmente aleatorio?

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Che rischio atteso avrebbe un classificatore totalmente aleatorio?
  90%

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Che rischio atteso avrebbe un classificatore totalmente aleatorio?
  90%

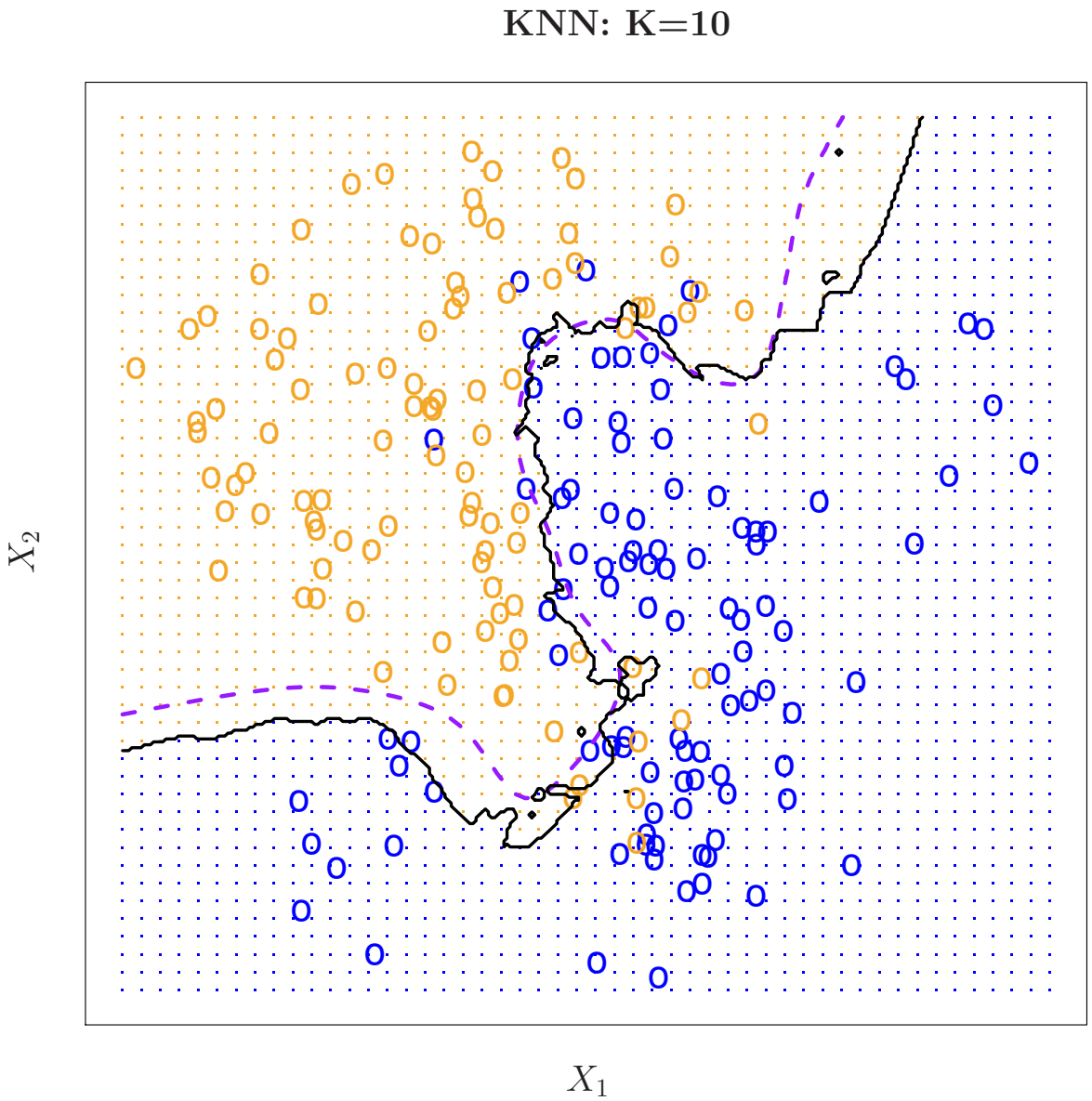
Esempi di classificazione errata:

Migliorare l'accuratezza di K-NN: scelta di K

Cosa succede variando K?

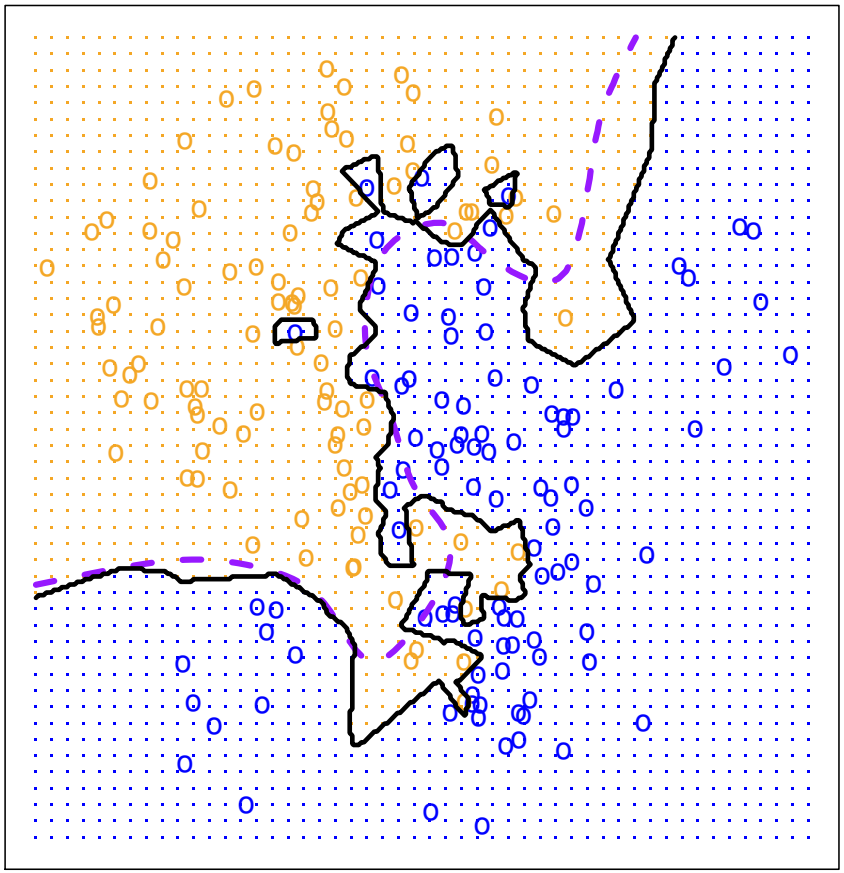
K	1	3	5	7	9	11
Errore di test	3.09%	2.94%	3.13%	3.10%	3.43%	3.34%

Effetto della variazione di K

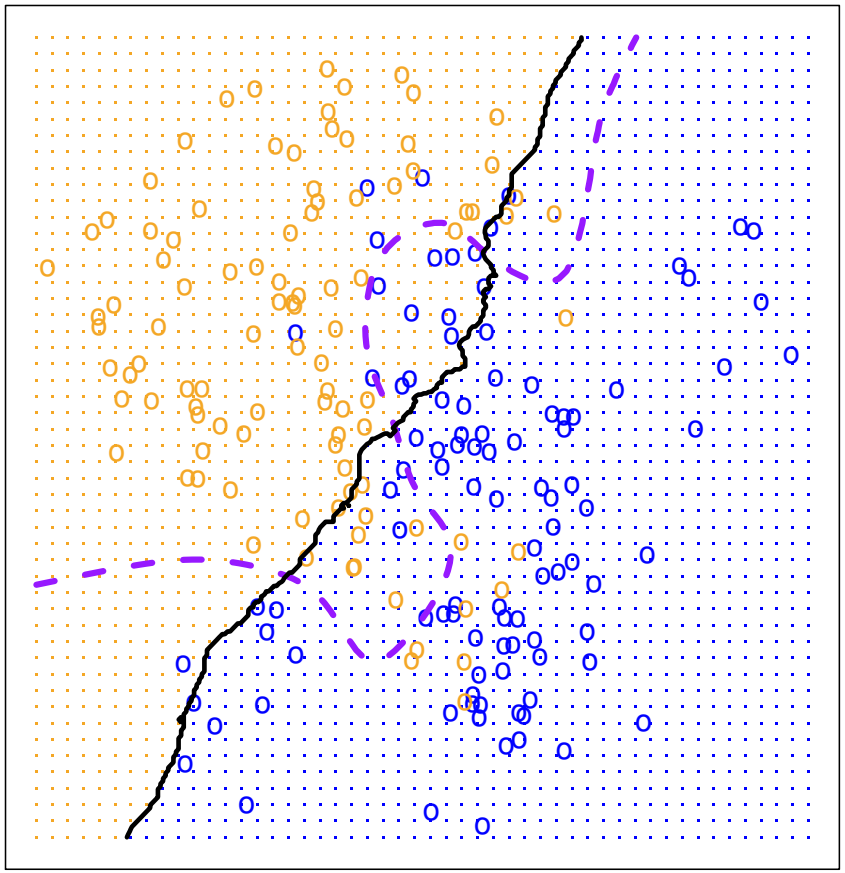


Effetto della variazione di K

KNN: K=1

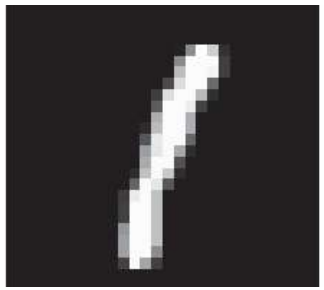
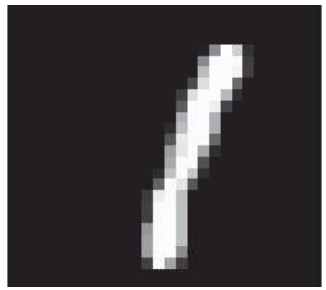


KNN: K=100



Migliorare l'accuratezza di K-NN: la funzione distanza

La distanza euclidea (ℓ_2) tra queste due immagini è molto alta!

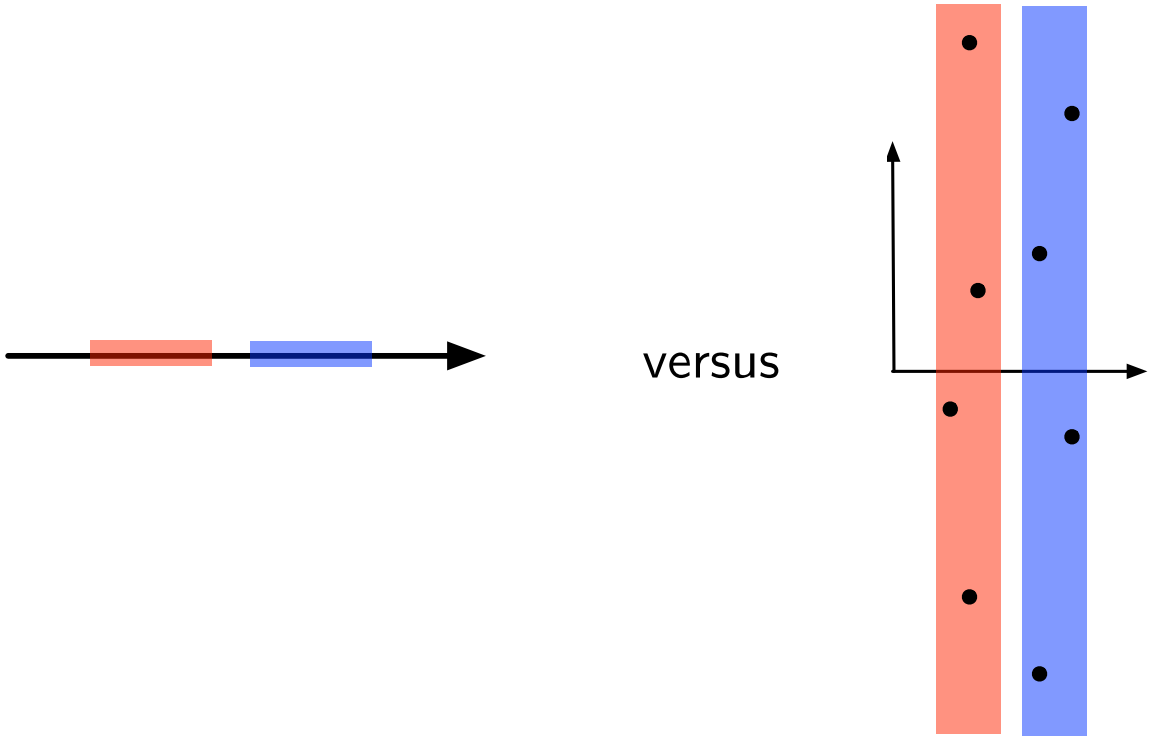
Idea migliore: usare funzioni distanza *invarianti* rispetto a:

- Piccole traslazioni e rotazioni: es. *tangent distance*
- Una classe più ampia di deformazioni naturali: es. *shape context*

Distanza	ℓ_2	tangent distance	shape context
Errore di test	3.09%	1.10%	0.63%

K-NN: L'impatto di variabili rumorose

Una buona *feature selection* è essenziale prima di applicare NN: anche solo una variabile poco significativa può avere effetti deleteri!



K-NN in scikit-learn

```
clf = KNeighborsClassifier(n_neighbors, metric)
clf.fit(X, y)
```

K-NN: Velocizzare la ricerca

Ricerca naïf dei K punti più vicini richiede tempo $m \cdot d$ per un dataset di taglia m su d variabili: lenta!

Esistono *strutture dati* che, preprocessando i dati, velocizzano la ricerca:

- K-d trees
- Ball trees
- Locality sensitive hashing

Spesso supportate dalle librerie di Machine Learning

Per esempio, `scikit-learn` offre le strutture `KDTree` e `BallTree`