k Nearest Neighbours Algorithm

Given:

- similarity metric (like cosine),
- parameter k,
- reference set $X = \{\vec{x}_1, \vec{x}_2, \ldots \vec{x}_m\}$,
- query $\vec{y}$,
- target classes, $c_1, \ldots c_d$, and assignments to X.

1. Initialise $L(c_j) := 0$ for each class j
2. For all $\vec{x}_i \in k$ closest training vectors to $\vec{y}$:
   - For each class c_{ji} to which $\vec{x}_i$ belongs:
     - $L(c_{ji})$ += $\text{sim}(\vec{x}_i, \vec{y})$ [or 1]
3. Choose c_j with largest $L(c_j)$

k Nearest Neighbours Algorithm

Advantages:

- no training phase,
- guaranteed error bounds (with enough data): when k=1, it converges to 2 x *Bayes error rate*
  - the optimal error rate attainable by maximising $P(c_j|\vec{y}, X)$
  - distances must also have been standardised (mean = 0, variance = 1)
- fairer weighting of evidence than cosine.

k Nearest Neighbours Algorithm

Disadvantages:

- must choose k,
- must choose similarity metric,
- time/space complexity not good: $\mathcal{O}(n \cdot |X|)$ ($n =$ dimension of $\vec{y}$ and $\vec{x}_i$),
- performance not good if variances of classes are different.

But there are fast approximations: choose k that are pretty close, but perhaps not closest (ϵ-NN).