Instance based learning

- Instance based learning simply store the data points without explicitly describing the target function
- Calculates the relation between the test instance and each training instance

- **Lazy learning:** It delays processing until a new instance comes in

- **Example:** KNN, Locally weighted regression

K-NN

Key idea: just store all training examples $\langle x_i, f(x_i) \rangle$

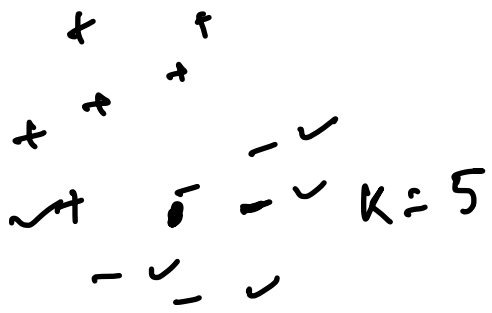
Nearest neighbor:

- Given query instance x_q, first locate nearest training example x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$

k-Nearest neighbor:

- Given x_q, take vote among its k nearest nbrs (if discrete-valued target function)
- take mean of f values of k nearest nbrs (if real-valued)

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$



- Instances map to points in $\Re^n$
- Less than 20 attributes per instance
- Lots of training data

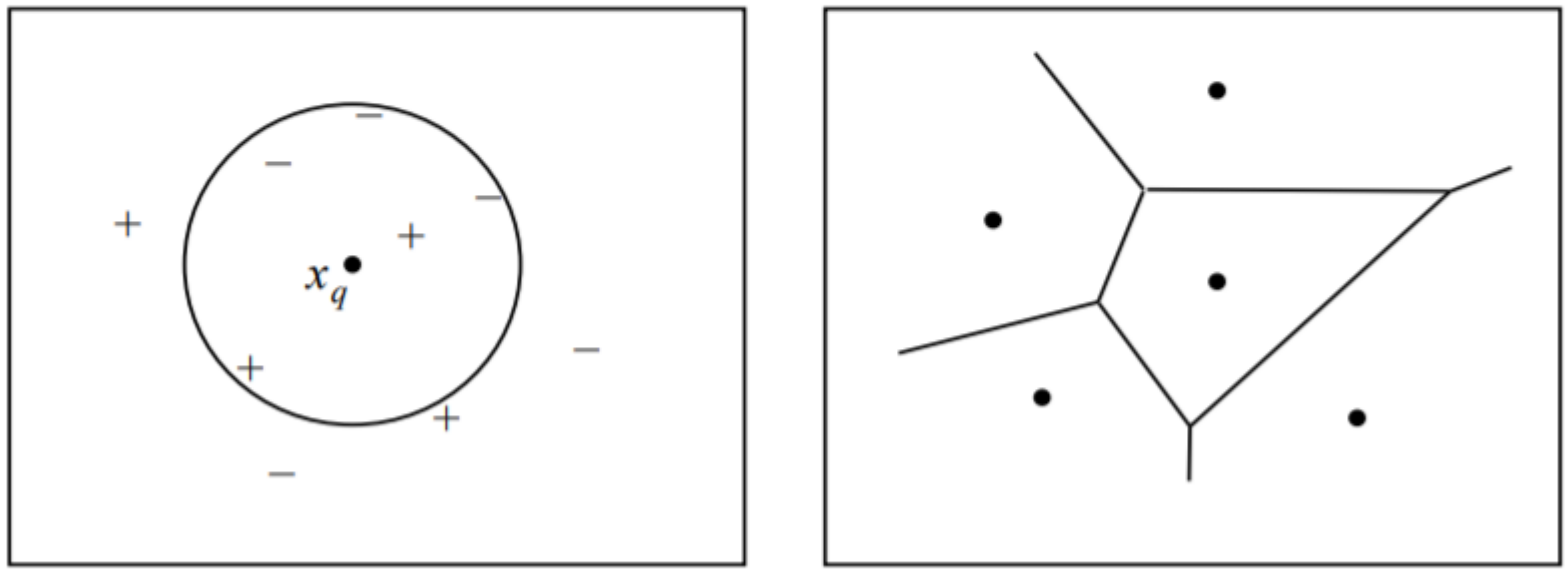
Advantages:

- Training is very fast
- Learn complex target functions
- Don't lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

Voronoi Diagram



The convex polygon surrounding each training example indicates the region of instance space closest to that point (i.e., the instances for which the 1-NEARESNT NEIGHBOR algorithm will assign the classification belonging to that training example).

Distance-Weighted kNN

Might want weight nearer neighbors more heavily...

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

and $d(x_q, x_i)$ is distance between x_q and x_i

Note now it makes sense to use *all* training examples instead of just k

$\rightarrow$ Shepard's method

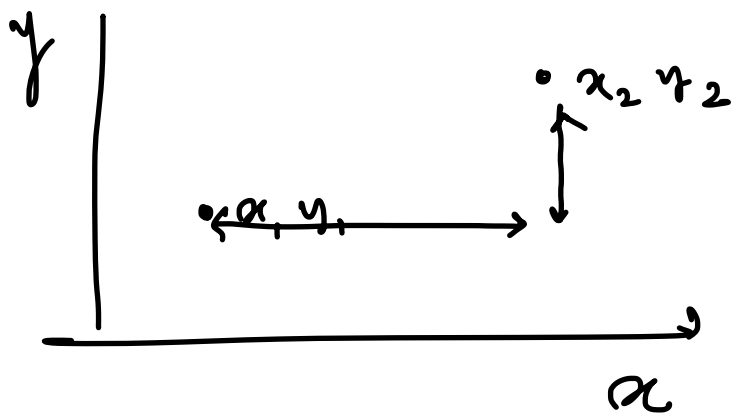
Distance metrics

$$dist(x,z) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p}$$

- Euclidean
- Manhattan
- Minkowski

$p = 1$

$$\sum_{r=1}^{d} \left(|x_r - z_r| \right)$$



$p \to \infty$

$$Dist_{\infty}(x,z) = \max_r |x_r - z_r|$$

Chebshev. distance

Time complexity of kNN

O(NM+kN)

N=# of training docs
M=Number of features

- Initialize $selected_i$=0 for all observations *i* in the training set
- For each training set observation i, compute $dist_i$, the distance from the new observation to training set observation *i*
- For *j*=1 to *k*: Loop through all training set observations, selecting the index *i* with the smallest $dist_i$ value and for which $selected_i$=0. Select this observation by setting $selected_i$=1

Return the k selected indices

Time complexity of kNN

O(NM)

N=# of training docs
M=No. of features

- For each training set observation *i*, compute *disti*, the distance from the new observation to training set observation *I*
- Run the quickselect algorithm to compute the kth smallest distance in O(N) runtime
- Return all indices no larger than the computed kth smallest distance

https://en.wikipedia.org/wiki/Quickselect

```
function quickSelect(list, left, right, k)

   if left = right
      return list[left]

   Select a pivotIndex between left and right

   pivotIndex := partition(list, left, right,
                                    pivotIndex)
   if k = pivotIndex
      return list[k]
   else if k < pivotIndex
      right := pivotIndex - 1
   else
      left := pivotIndex + 1
```

kNN: Memory/instance based learning

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
  - We always preprocess the training set, so in reality training time of kNN is linear.
- kNN is very accurate if training set is large.
- But kNN can be very inaccurate if training set is small.

Decision surface

Bias and variance trade off

Why ML needs large dataset

Curse of dimensionality

KNN is a non-parametric model

Disadvantages

- Cost of classifying new instances can be high
  - Nearly all computation takes place at classification time rather than when the training examples are first encountered
  - Efficiently indexing training examples

- Consider ***all*** attributes of the instances when attempting to retrieve similar training examples from memory
  - If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.

Assumption: Similar inputs have similar outputs.

Test point: x

S_x: the set of the k nearest neighbors of x.

$S_x \subseteq D \quad |S_x| = K$ D: Dataset

$\forall (x', y') \in D \setminus S_x$

$$\text{distance}(x, x') \geq \max_{(x'', y'') \in S_x} \text{distance}(x, x'')$$

Classifier $h(x) = \text{mode}(\{ y'' : (x'', y'') \in S_x \})$

where mode $(\cdot)$ means to select the label of the highest occurrence.

Bayes Optimal classifier

$p(y|x)$ $\quad p(+|x) = .8 \quad p(-|x) = .2$

$$y^* = h_{opt}(x) = \arg\max_{y} p(y|x)$$

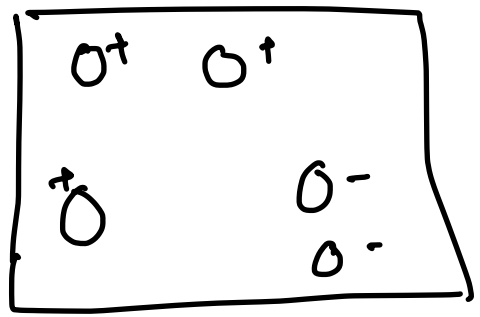
$$\epsilon_{Bayes\,opt} = 1 - p(y^*|x)$$

← lower bound of error rate

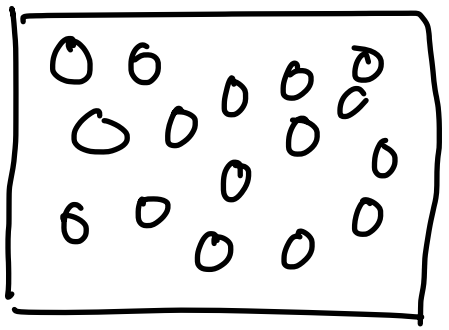
← lower error.

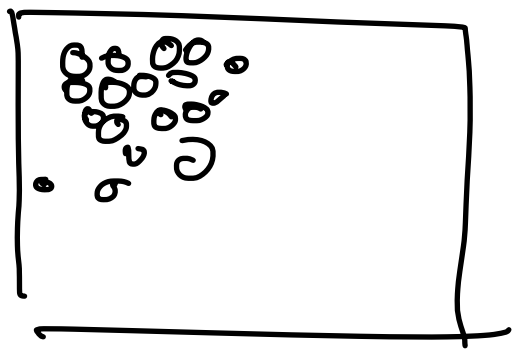
Constant prediction

Thm As $n \to \infty$, the 1-NN error is no more than twice the error of the Bayes Optimal classifier.



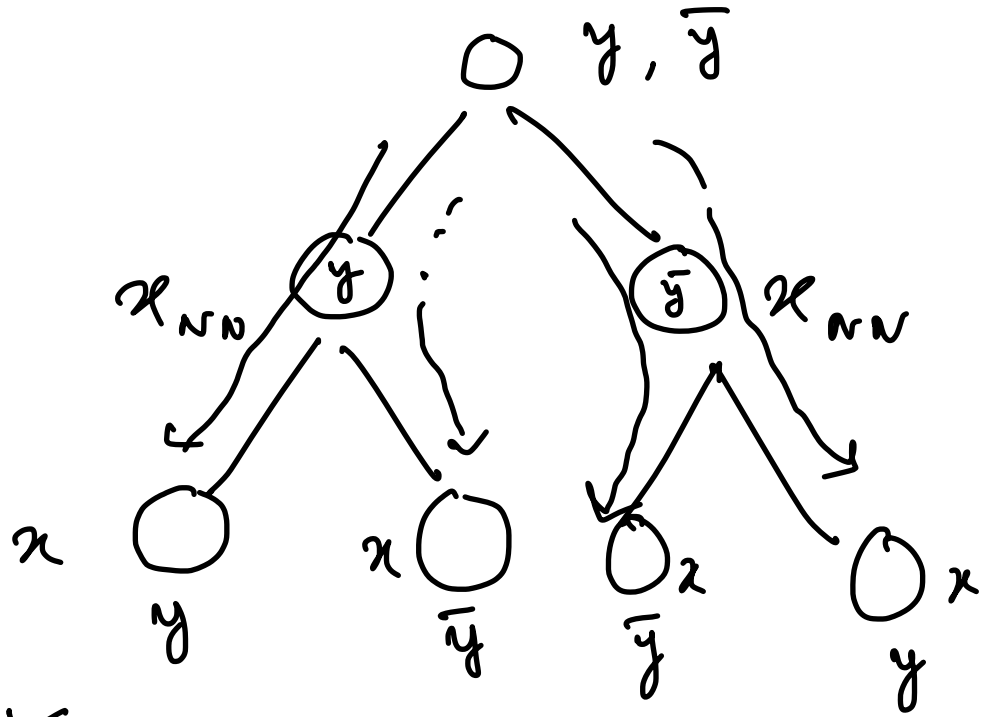
$n = 5$





x_{NN} ← nearest neighbour of x

$n \to \infty$ dist $(x_{NN}, x) \to 0 \Rightarrow x_{NN} \to x$



x_{NN}

x

$$\epsilon_{NN} = P(y^* | x_{NN})(1 - P(y^* | x)) + P(y^* | x)(1 - P(y^* | x_{NN}))$$

$$\leq (1 - P(y^* | x)) + (1 - P(y^* | x_{NN}))$$

$$= 2(1 - P(y^* | x))$$

$\leq 2 \in_{Bayopt}$

$p(y^* | x) = p(y^* | x_{nn})$

— Curse of dimensionality —

in higher dimensions space. Points that are drawn from a prob dist., tend to never be close together.