K-nearest neighbours

Javier Béjar

LSI - FIB

Term 2012/2013

classifier

supervised (not this class)

$f(\text{datapoint\_representation}) \simeq \text{label}$

$x_i \qquad y_i$

Parametric vs Non parametric Models

- In the models that we have seen, we select a hypothesis space and adjust a fixed set of parameters with the training data ($h_\alpha(x)$)
- We assume that the parameters α summarize the training and we can forget about it
- This methods are called **parametric** models
- When we have a small amount of data it makes sense to have a small set of parameters and to constraint the complexity of the model (avoiding overfitting)

Parametric vs Non parametric Models

- When we have a large quantity of data, overfitting is less an issue
- If data shows that the hipothesis has to be complex, we can try to adjust to that complexity
- A **non parametric** model is one that can not be characterized by a fixed set of parameters
- A family of non parametric models is **Instance Based Learning**

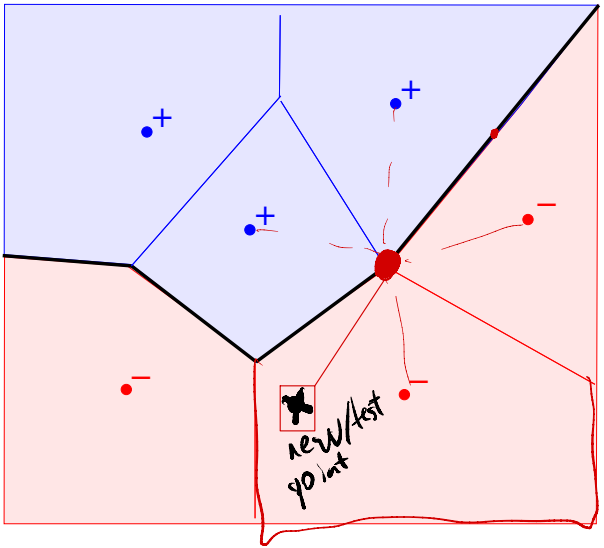
Instance Based Learning

- Instance based learning is based on the memorization of the dataset
- The number of parameters is unbounded and grows with the size of the data
- There is not a model associated to the learned concepts
- The classification is obtained by looking into the memorized examples
- The cost of the learning process is 0, all the cost is in the computation of the prediction
- This kind learning is also known as **lazy learning**

K-nearest neighbours

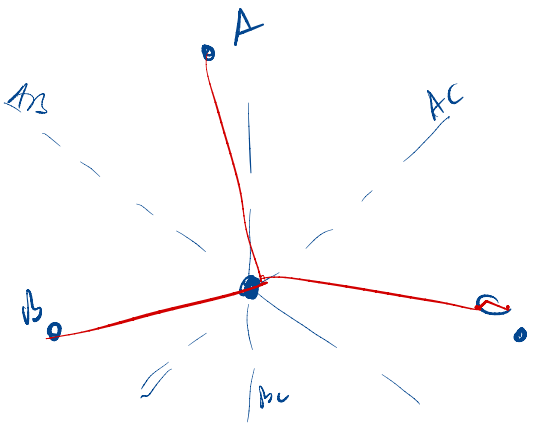
- **K-nearest neighbours** uses the local neighborhood to obtain a prediction
- The K memorized examples more similar to the one that is being classified are retrieved
- A distance function is needed to compare the examples similarity
  - Euclidean distance ($d(x_j, x_k) = \sqrt{\sum_i (x_{j,i} - x_{k,i})^2}$)
  - Mahnattan distance ($d(x_j, x_k) = \sum_i |x_{j,i} - x_{k,i}|$)
- This means that if we change the distance function, we change how examples are classified

K-nearest neighbours - hypothesis space (1 neighbour)



$\{+ \; -\}$
training set
(I know labels)

unique partition of plane (the space s.t $\forall x$ is in the partition with closest centroid.



K-nearest neighbours - Algorithm

Neighbourhood-k
$N(x_{new})$ = the closest K points in training

- Training: Store all the examples
- Prediction: $h(x_{new})$
  - Let be $x_1, \ldots, x_k$ the k more similar examples to x_{new}
  - $h(x_{new})$= combine_predictions($x_1, \ldots, x_k$)
- The parameters of the algorithm are the number k of neighbours and the procedure for combining the predictions of the k examples
- The value of k has to be adjusted (crossvalidation)
  - We can overfit (k too low)
  - We can underfit (k too high)

k=3

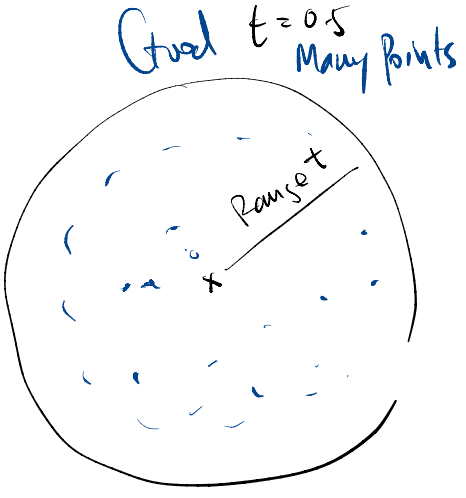
Good - close

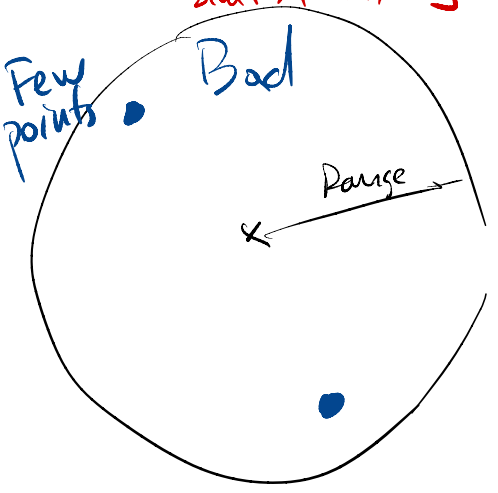
Bad - far away

- data density (not uniform)
  - easy to find close neighbors (dense / typical)
  - hard to find close neighbors (sparse / anomaly)

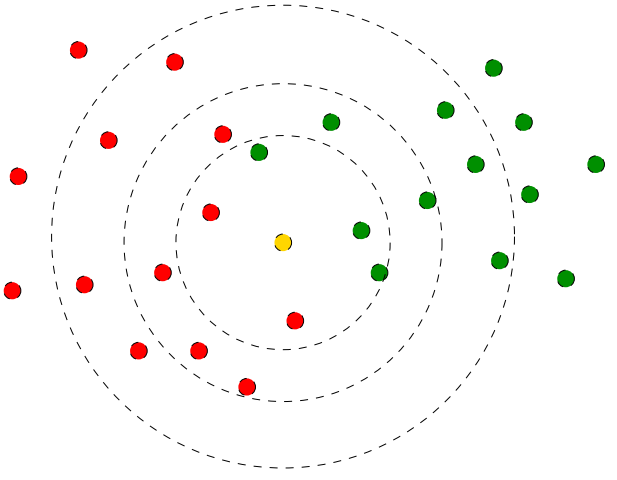
$N_x = \{$ closest k points to $x \}$

$= \{ z \in \text{training} \mid sim(x,z) \geq t \}$ within t similarity





K-nearest neighbours - Prediction



Weighted version

- $f(x,z) = similarity(x,z)$
- use all points

$$predict(x, L) = \text{avg} \sum_{i=1}^{N} L(z_i) \cdot f(x, z_i)$$

weighted avg

x: test, L: label

$i = 1$ over train set

$L(z_i)$: label L for point z_i

Looking for neighbours

- Looking for the K-nearest examples for a new example can be expensive
- The straightforward algorithm has a cost $O(n \log(k))$, not good if the dataset is large
- We can use indexing with *k-d trees* (multidimensional binary search trees)
  - They are good only if we have around 2^{dim} examples, so not good for high dimensionality
- We can use *locality sensitive hashing* (approximate k-nn)
  - Examples are inserted in multiple hash tables that use hash functions that with high probability put together examples that are close
  - We retrieve from all the hash tables the examples that are in the bin of the query example
  - We compute the k-nn only with these examples

K-nearest neighbours - Variants

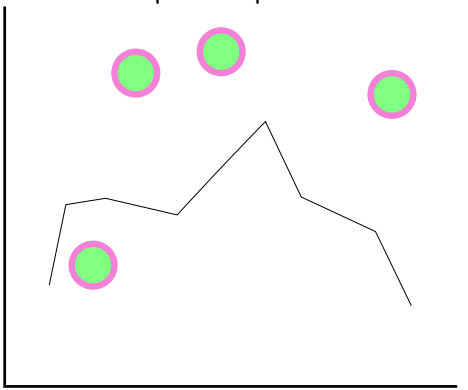
- There are different possibilities for computing the class from the k nearest neighbours
  - Majority vote
  - Distance weighted vote
    - Inverse of the distance
    - Inverse of the square of the distance
    - Kernel functions (gaussian kernel, tricube kernel, ...)
- Once we use weights for the prediction we can relax the constraint of using only k neighbours
  1. We can use k examples (local model)
  2. We can use all examples (global model)

K-nearest neighbours - Regression

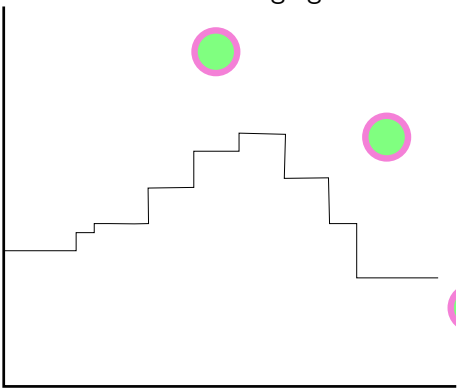
- We can extend this method from classification to regression
- Instead of combining the discrete predictions of k-neighbours we have to combine continuous predictions
- This predictions can be obtained in different ways:
  - Simple interpolation
  - Averaging
  - Local linear regression
  - Local weighted regression
- The time complexity of the prediction will depend on the method

K-nearest neighbours - Regression

Simple interpolation

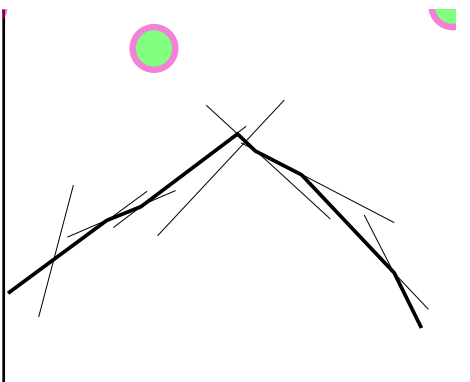


K-nn averaging



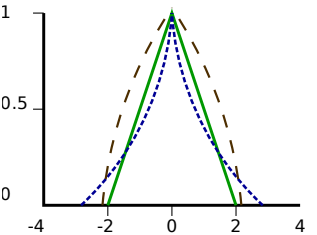
K-nearest neighbours - Regression (linear)

- K-nn linear regression fits the best line between the neighbors
- A linear regression problem has to be solved for each query (least squares regression)



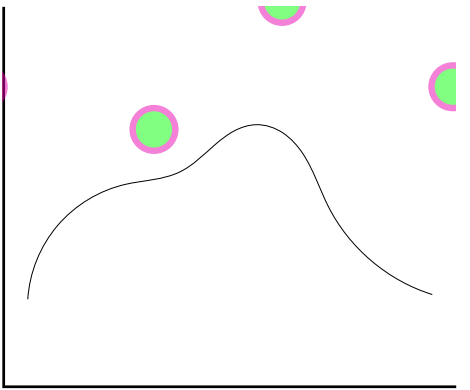
K-nearest neighbours - Regression (LWR)

- Local weighted regression uses a function to weight the contribution of the neighbours depending on the distance, this is done using a **kernel function**



- Kernel functions have a width parameter that determines the decay of the weight (it has to be adjusted)
  - Too narrow $\Longrightarrow$ overfitting
  - Too wide $\Longrightarrow$ underfitting
- A weighted linear regression problem has to be solved for each query (gradient descent search)

K-nearest neighbours - Regression (LWR)



K-nearest neighbours - Advantages

- The cost of the learning process is zero
- No assumptions about the characteristics of the concepts to learn have to be done
- Complex concepts can be learned by local approximation using simple procedures

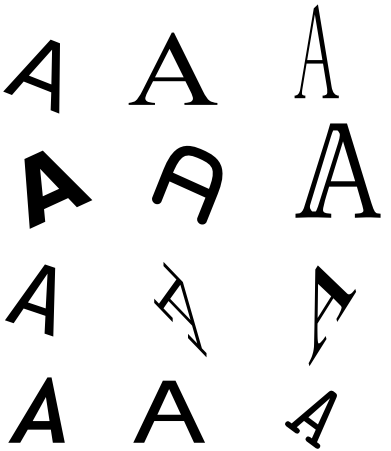
K-nearest neighbours - Drawbacks

- The model can not be interpreted (there is no description of the learned concepts)
- It is computationally expensive to find the k nearest neighbours when the dataset is very large
- Performance depends on the number of dimensions that we have (*curse of dimensionality*) $\Longrightarrow$ *Attribute Selection*

The Curse of dimensionality

- The more dimensions we have, the more examples we need to approximate a hypothesis
- The number of examples that we have in a volume of space decreases exponentially with the number of dimensions
- This is specially bad for k-nearest neighbors
  - If the number of dimensions is very high the nearest neighbours can be very far away

Optical Character Recognition



- OCR capital letters
- 14 Attributes (All continuous)
- Attributes: horizontal position of box, vertical position of box, width of box, height of box, total num on pixels, mean x of on pixels in box, ...
- 20000 instances
- 26 classes (A-Z)
- Validation: 10 fold cross validation

Optical Character Recognition: Models

- K-nn 1 (Euclidean distance, weighted): accuracy 96.0%
- K-nn 5 (Manhattan distance, weighted): accuracy 95.9%
- K-nn 1 (Correlation distance, weighted): accuracy 95.1%