Nearest Neighbors

Nearest Neighbor

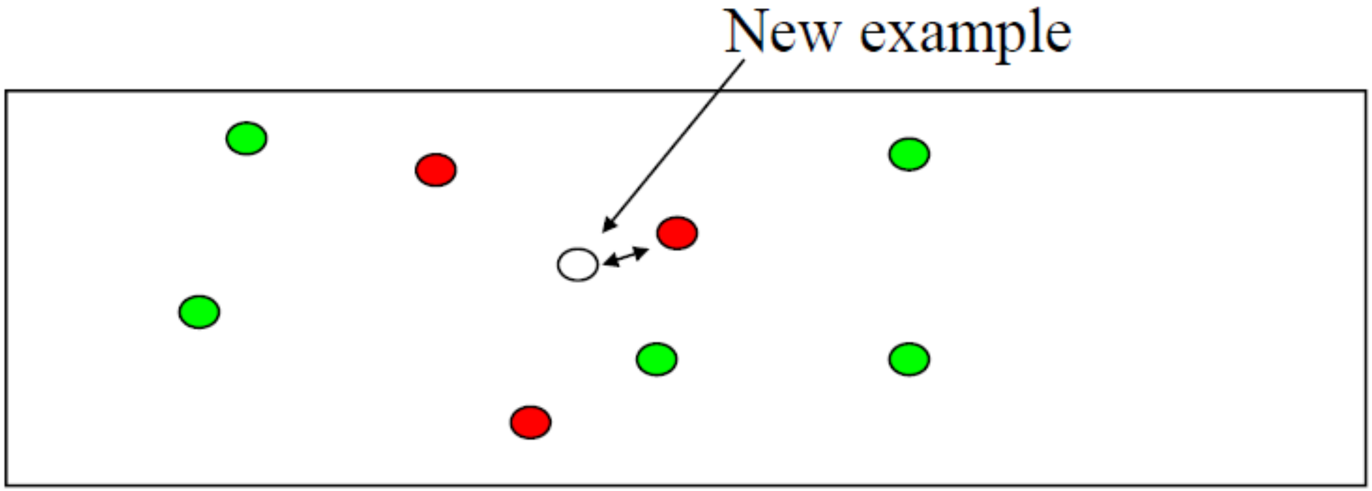
- Hypothesis space
  - Variable size
  - Deterministic
  - Continuous parameters
- Algorithm
  - Direct computation
  - Lazy

Nearest Neighbor Algorithm

- Our first lazy algorithm
  - The learning does not occur till the test example is presented
  - In contrast to the “eager” algorithms (those algorithms that carry out learning without seeing the test example and discard the training examples after learning)

Nearest Neighbor Algorithm

- Remember all training examples
- Given a new example **x**, find the its closest training example <**x**^{i}, y^i> and predict y^i



Nearest Neighbor Algorithm

- Classify a new example **x**, by finding the training example $<\mathbf{x}_i,y_i>$ that is nearest to **x** according to Eucledian distance

$$\| \mathbf{x} - \mathbf{x}_i \| = \sqrt{\sum_i (x_j - x_{ij})^2}$$

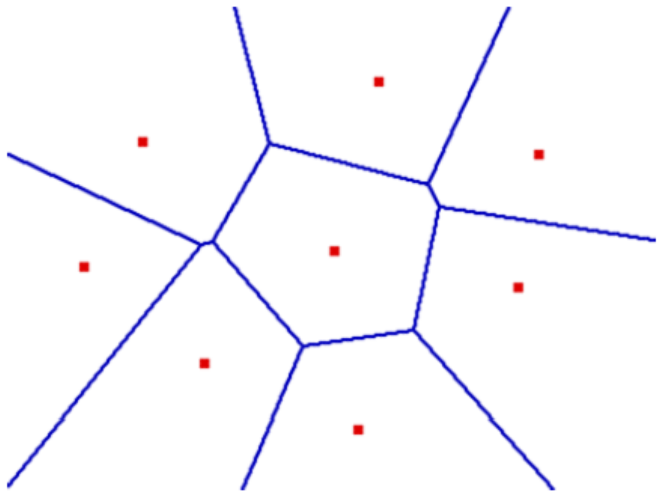
and guess the class $\hat{y} = y_i$

- For efficiency, we could simply use the squared distance and get the same answer by avoiding the square root computation

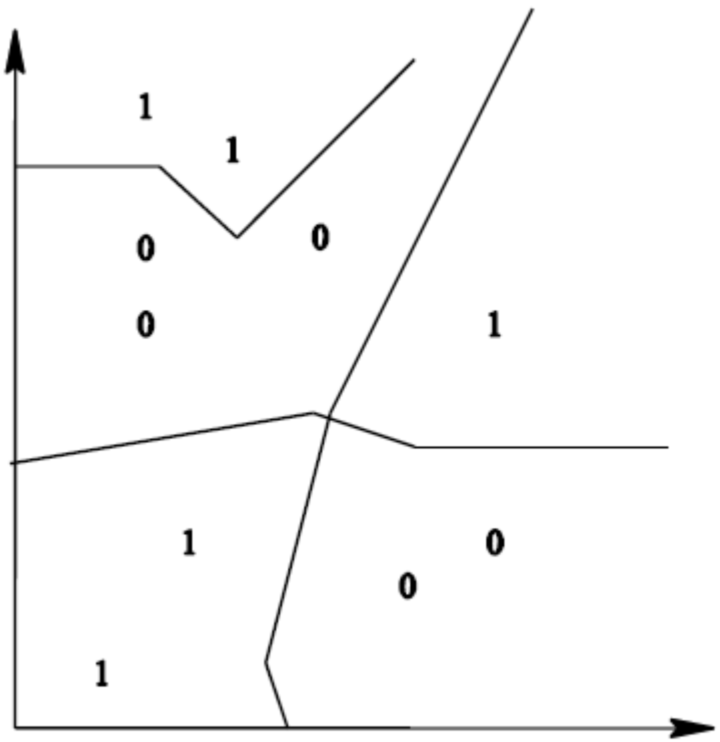
$$\| \mathbf{x} - \mathbf{x}_i \|^2 = \sum_i (x_j - x_{ij})^2$$

Decision Boundaries: The Voronoi Diagram

- Voronoi Diagram: Given a set of points, it describes the areas that are nearest to any given point
- These areas can be viewed as zones of control
- This is also same as the post office problem

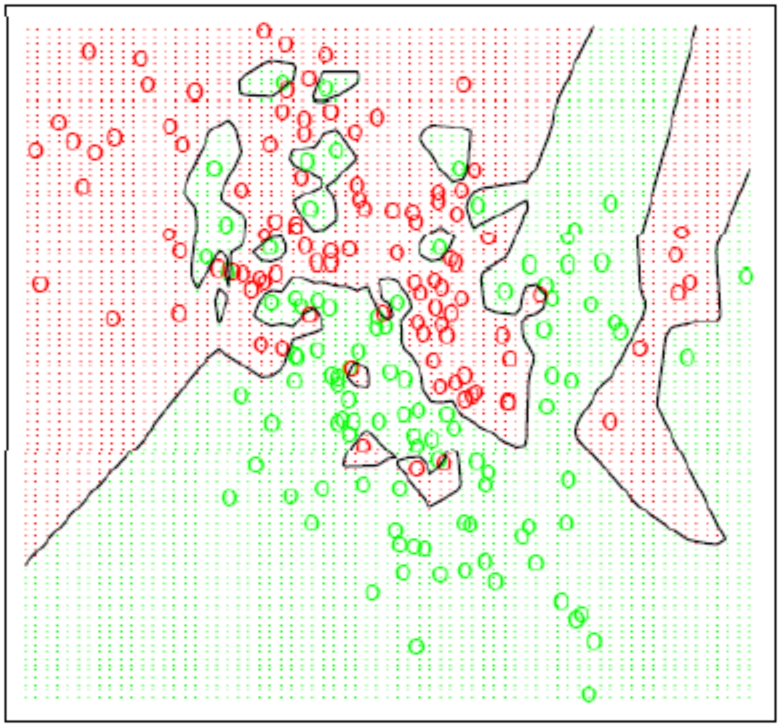


Decision Boundaries



- Decision boundaries are formed by the **current subset** of the examples considered
- Each line segment is equidistant between two points of **opposite** class
- If you consider more examples, the decision boundaries can become more complex
- Complexity of the boundary increases with the number of examples considered

Boundaries



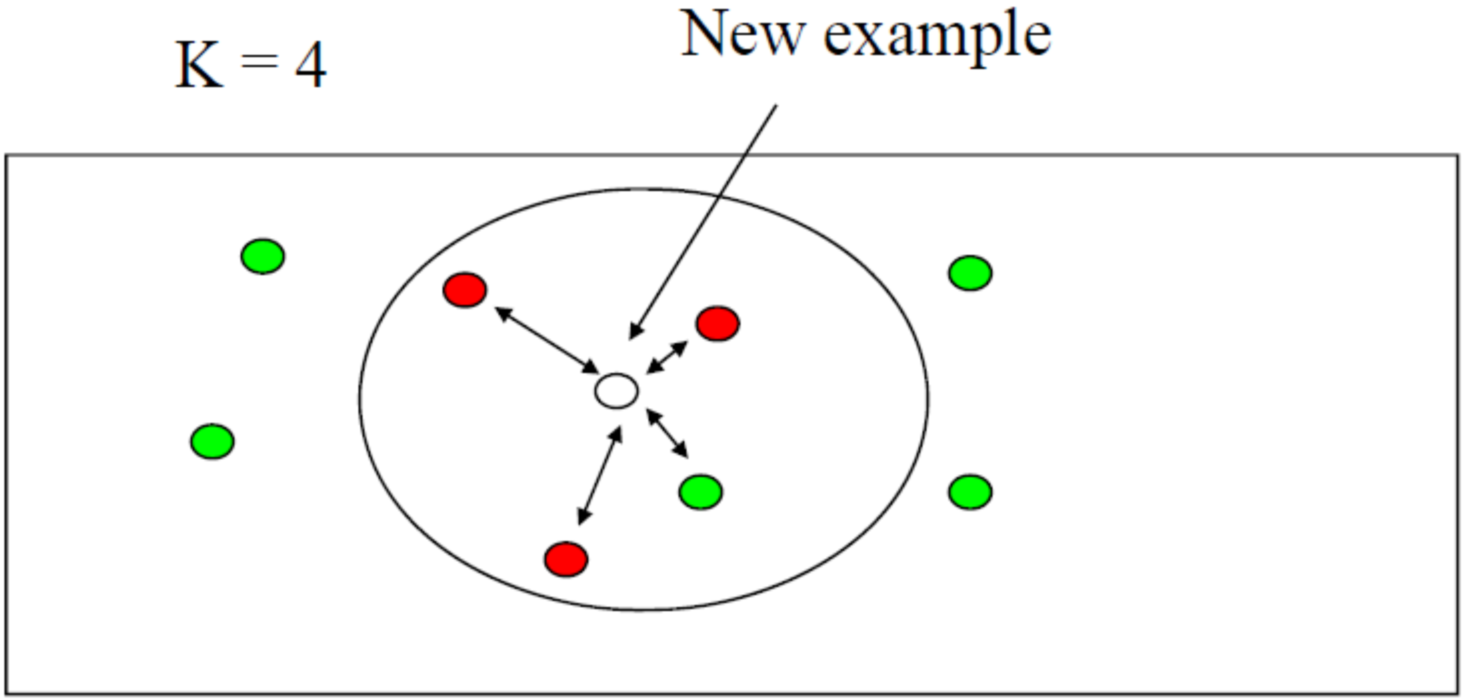
- Noise and large number of examples can easily lead to over fitting (as we could start having these islands of neighborhoods)

NN depends critically on the distance metric

- Normalize Feature Values
  - All feature values must have the same range of values. Otherwise, features will larger range become more important
- Sensitive to Irrelevant inputs
  - Irrelevant or noisy features will add random perturbations to the distance measure and can easily hurt performance
- Learn a distance metric:
  - One approach: Weigh each feature based on its mutual information to the target class. Then use the weighted square distance as the distance metric $\sum_i w_i (x_j - x_{ij})^2$
  - Alternatively use the Mahalanobis distance
- Smoothing:
  - Find the *k* nearest neighbors and have them vote. This is one good way to reduce the effect of noise in the labels

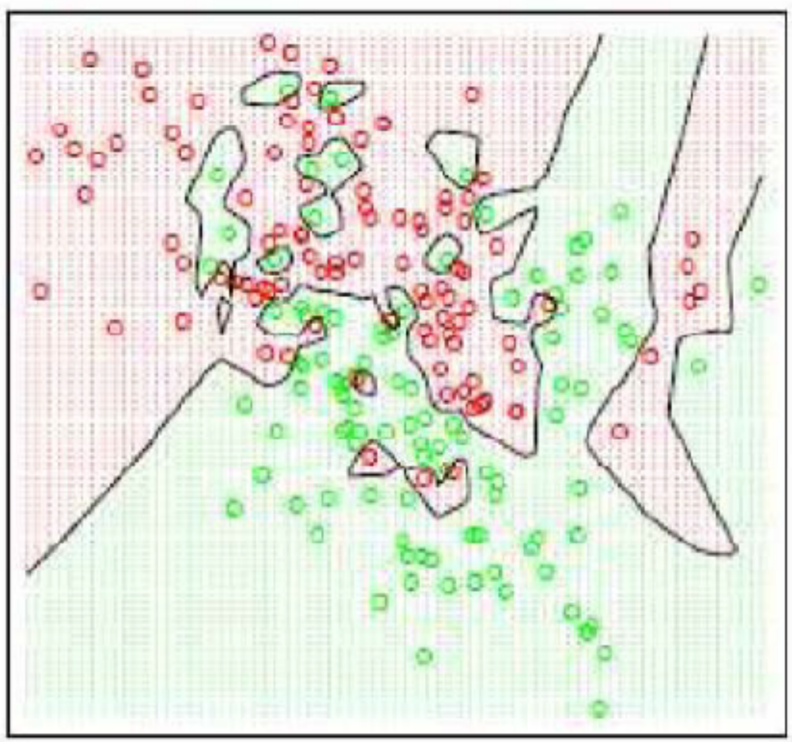
K-Nearest Neighbor

Example:

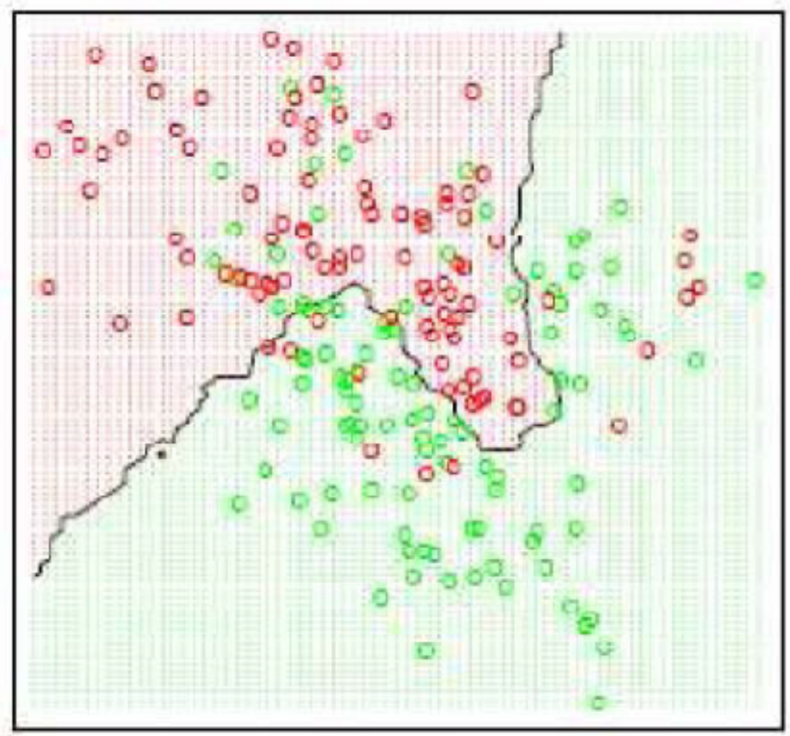


Effect of K

K=1



K=15



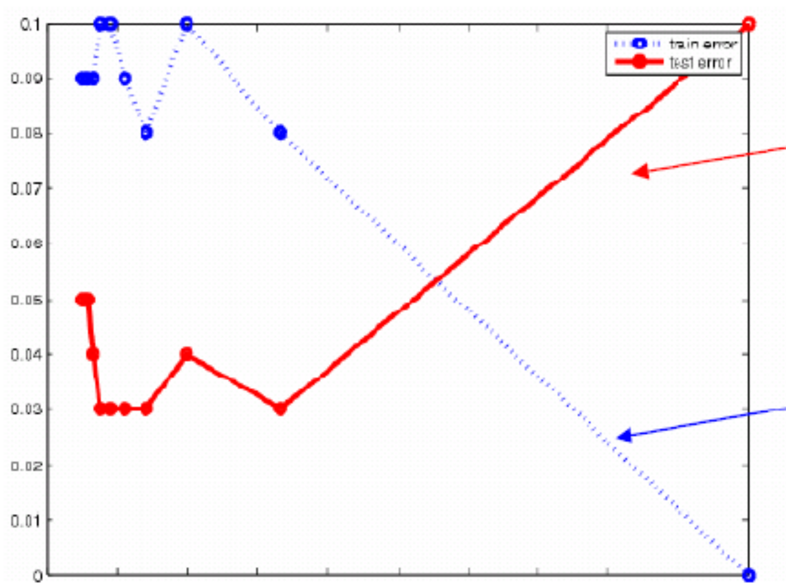
Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Larger k produces smoother boundary effect and can reduce the impact of class label noise.

But when K = N, we always predict the majority class

Overfitting is easily possible

- Can we choose k to minimize the mistakes that we make on training examples (*training error*)?



Test error

Train error

K=20

K=1

Model complexity

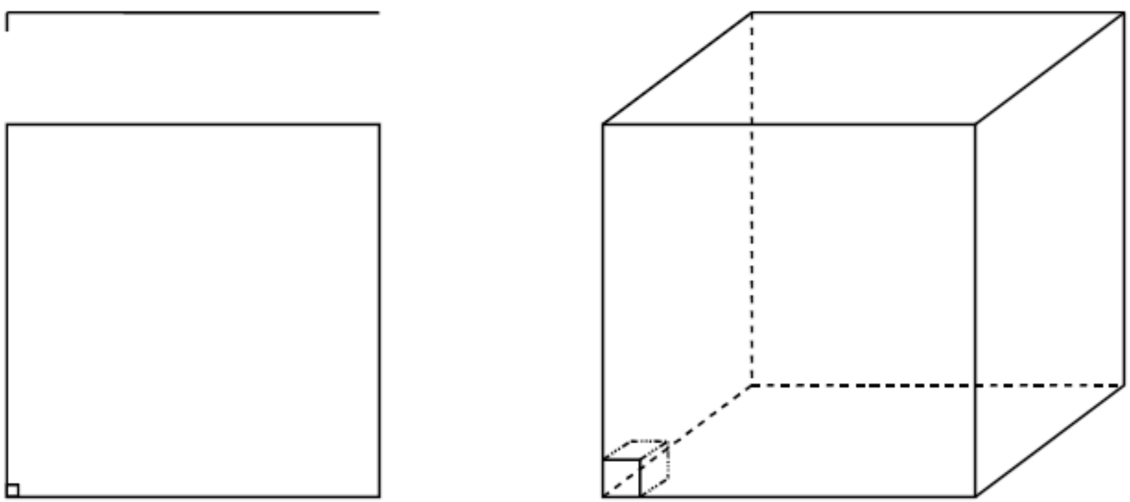
A model selection problem that we will study later

Distance Weighted Nearest Neighbor

- It makes sense to weight the contribution of each example according to the distance to the new query example
  - Weight varies inversely with the distance, such that examples closer to the query points get higher weight
- Instead of only k examples, we could allow all training examples to contribute
  - Shepard's method (Shepard 1968)

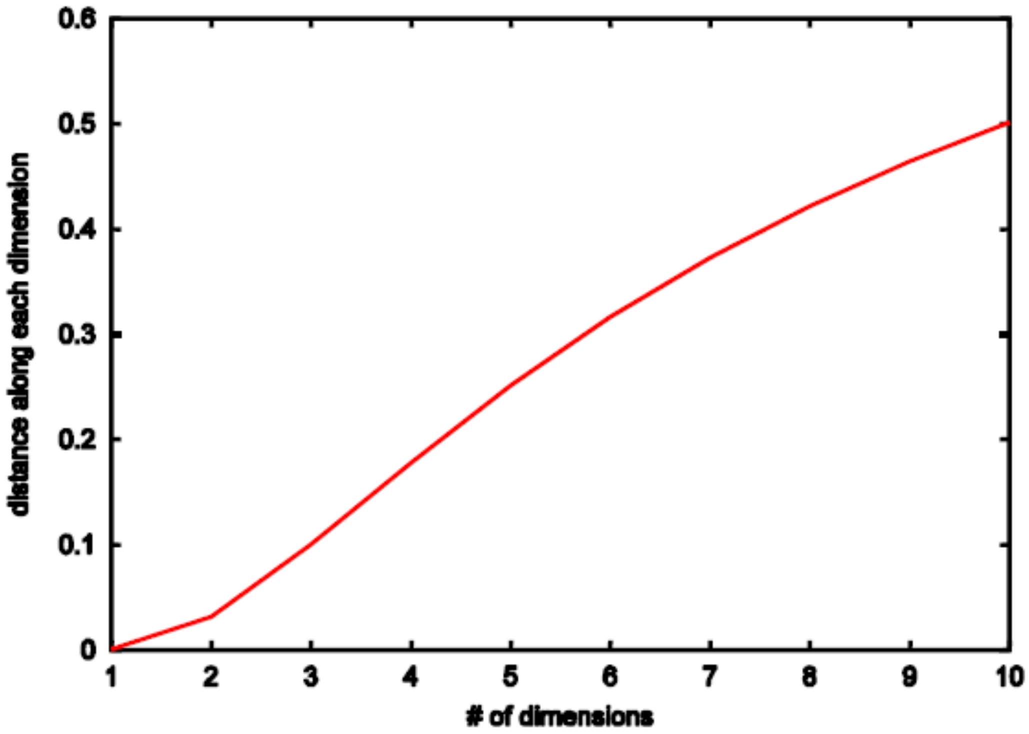
Curse of Dimensionality

- *k*NN breaks down in high-dimensional space
  - "Neighborhood" becomes very large.
- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-nn. Suppose our query point is at the origin.
  - In 1-dimension, we must go a distance of 5/5000 = 0.001 on the average to capture 5 nearest neighbors
  - In 2 dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume.
  - In d dimensions, we must go $(0.001)^{1/d}$



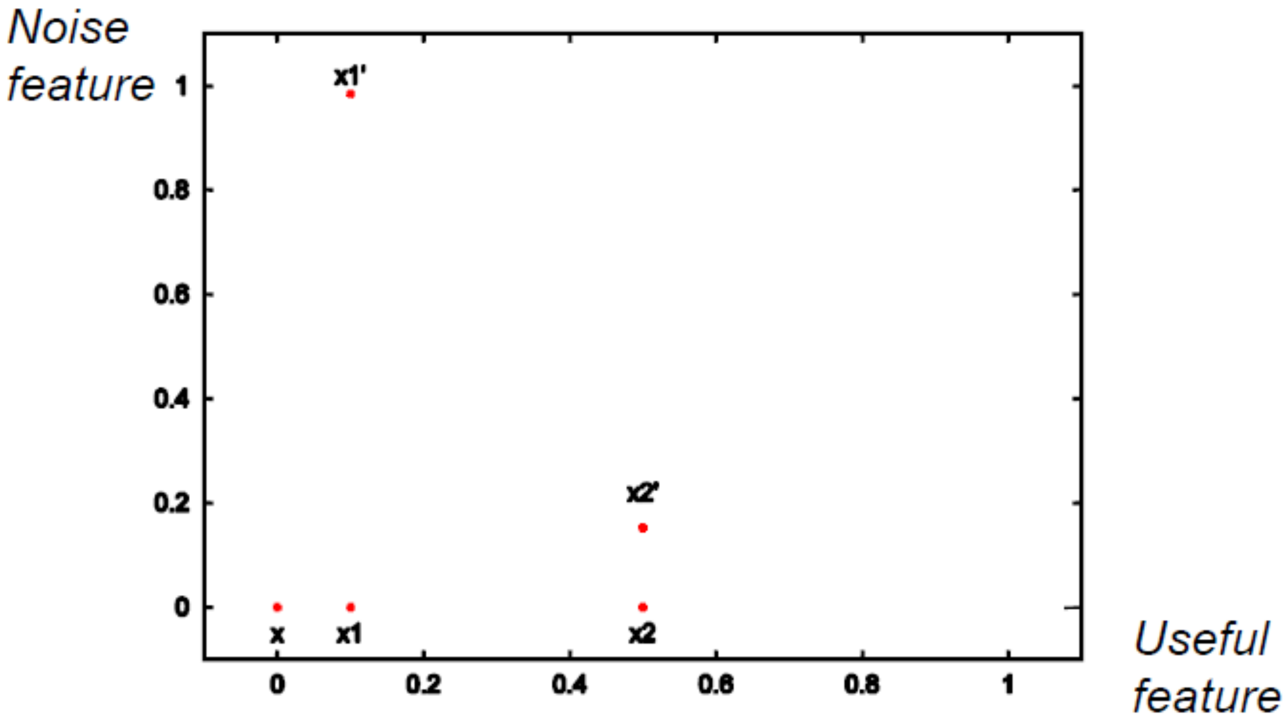
Curse of Dimensionality (2)

- With 5000 points in 10 dimensions, we must go 0.501 distance along each dimension in order to find the 5 nearest neighbors



Curse of Noisy/Irrelevant Features

- NN also breaks down when data contains irrelevant/noisy features.
- Consider a 1-d problem where query x is at the origin, our nearest neighbor is x_1 at 0.1, and our second nearest neighbor is x_2 at 0.5.
- Now add a uniformly random noisy feature.
  - $P(||x_2' - x||<||x_1' - x||) \approx 0.15$.



Problems of k_NN

- Nearest neighbor is easily misled by noisy/irrelevant features
- One approach: Learn a distance metric:
  - that weights each feature by its ability to minimize the prediction error, e.g., its mutual information with the class.
  - that weights each feature differently or only use a subset of features and use cross validation to select the weights or feature subsets
  - Learning distance function is an active research area

Sample Experimental Results

(see UCI archive for more)

Testbed	Testset Correctness		
	1-NN	D-Trees	Neural Nets
Wisconsin Cancer	<u>98%</u>	95%	96%
Heart Disease	<u>78%</u>	76%	?
Tumor	37%	38%	?
Appendicitis	83%	85%	86%

Summary of Nearest Neighbor

- Advantages
  - Learning is extremely simple and intuitive,
  - Very flexible decision boundaries
  - Variable-sized hypothesis space
- Disadvantages
  - distance function must be carefully chosen or tuned
  - irrelevant or correlated features have high impact and must be eliminated
  - typically cannot handle high dimensionality
  - computational costs: memory and classification-time computation
    - To reduce the cost of finding nearest neighbors, use data structure such as kd-tree

Criterion	**Perceptron**	**Logistic**	**LDA**	**DT**	**K-NN**
Mixed data	N	N	N	Y	N
Missing values	N	N	Y	Y	Some what
Outliers	N	Y	N	Y	Y
Monotone	N	N	N	Y	N
Scalability	Y	Y	Y	Y	N
Irrelevant i/p	N	N	N	Some what	N
Interpretable	Y	Y	Y	Y	N
Accurate	Y	Y	Y	N	N