CSE 537 Fall 2015

LEARNING FROM EXAMPLES: NON-PARAMETRIC MODELS
AIMA CHAPTER 18.8

- Instructor: Sael Lee

Slides are mostly made from AIMA resources

PARAMETRIC MODELS

- Parameter models: A learning model that summarizes data with a set of parameters of **fixed size.**
  - Use the training data to learn the fixed set of model parameter **w,** that completely describes our hypothesis $h_{\mathbf{w}}(\mathbf{x})$.
  - Model is fixed so the only **w** needed to be kept and all else (data etc.) can be thrown away.
  - the parameters have fixed size independent of number of training examples.
- Limitations of parametric models:
  - The model is fixed and have less freedom in describing the non-canonical state space.

NONPARAMETRIC MODELS

- Nonparametric model cannot be characterized by a bounded set of parameters.
- Can’t assume a model for distribution densities
- Might be a very complicated model with large number of parameters
- Nonparametric estimation principle: “Similar inputs have similar outputs”
  - Find similar data instances in the training data and interpolate/average their outputs

NONPARAMETRIC ESTIMATION

- Parametric estimation: all data instances affect the final glo bal estimate
  - Global methods
- Non-parametric estimation
  - No single global model
  - Local models are created as needed
  - Affected only by near-by instances
- Called Instance-based learning (memory-based learning)
  - Let the data speak for it self.
  - # of parameters may grow with the size of training examples
  - Ex> hypothesis that retains all the training examples and uses them to do prediction on new data

MEMORY-BASED METHOD

- Lazy method
  - Store training data of size N
  - O(N) memory
  - O(N) search to for similar data
- Parametric methods
  - d parameters, d<N
  - O(d) memory and processing

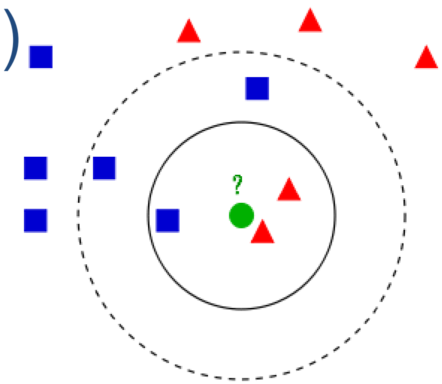
TABLE LOOKUP

- Howto:
  1. use all the training example to make lookup table
  2. if new data come, look it up in the table
  3. if the new data is in the lookup table, you have the answer.
- This does not generalize well
  - If the lookup table does not have the match for the new data, it will need to output default value.

K-NEAREST NEIGHBOR MODELS

Improving the table lookup: k-nearest neighbor lookup

- Given a query $\mathbf{x}_q$, find the k examples that are nearest to $\mathbf{x}_q$: kNN(k,$\mathbf{x}_q$)

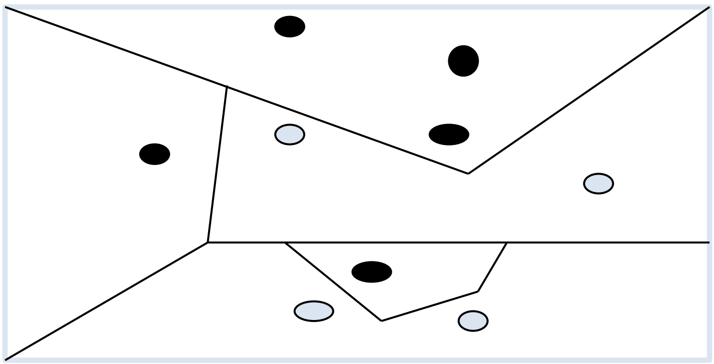


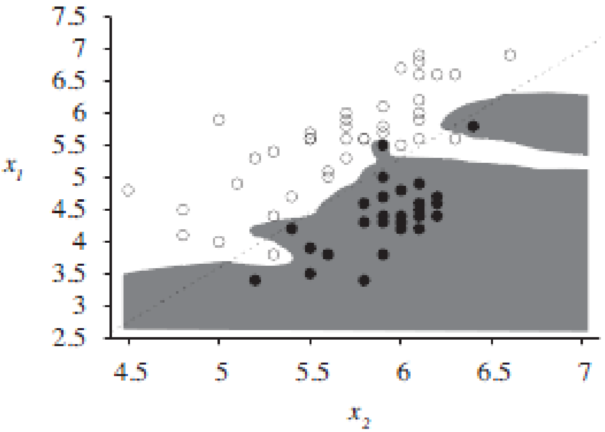
$$d_1(x) \le d_2(x) \le \cdots \le d_N(x)$$

- **Classification**: use the kNN(k,$\mathbf{x}_q$) and take the (weighted) plurality vote.
- Regression: use the kNN(k,$\mathbf{x}_q$)
  - take the mean or median
  - Or solve a linear regression problem on the neighbors.

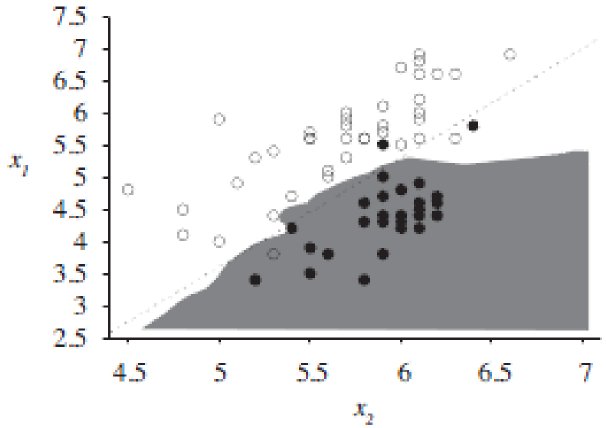
VORONOI DIAGRAM & EFFECT OF K

- Decision surface formed by the training examples





$(k = 1)$



$(k = 5)$

Fig. decision boundary of KNN for k=1 and k=5. low k value are subject to over fitting and high k values are subjected to under fitting. Need to learn *k*

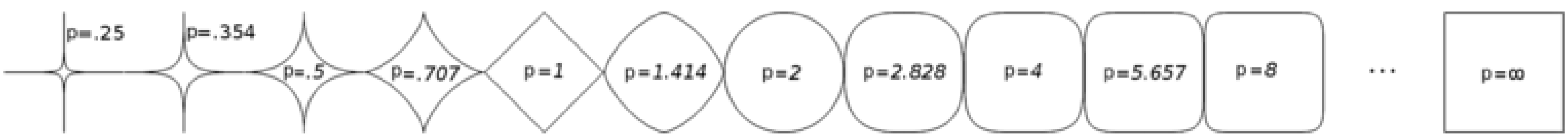
DEFINING "NEAREST" IN KNN

Definition of distance is important factor in accuracy.

- Minkowski distance L^p norm where $1 \leq p \leq \infty$

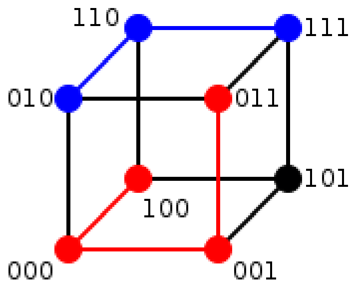
$$\left(\sum |x_i - y_i|^p \right) \quad .$$

  - p=1 : Manhattan distance
  - p=2 : Euclidean distance
  - Scale variant -> each feature should be normalized to have same scales.



DEFINING "NEAREST" IN KNN CONT.

- **Hamming distance**: number of positions at which the corresponding symbols are different in two strings of equal length.
  - measures the minimum number of substitutions required to change one string into the other



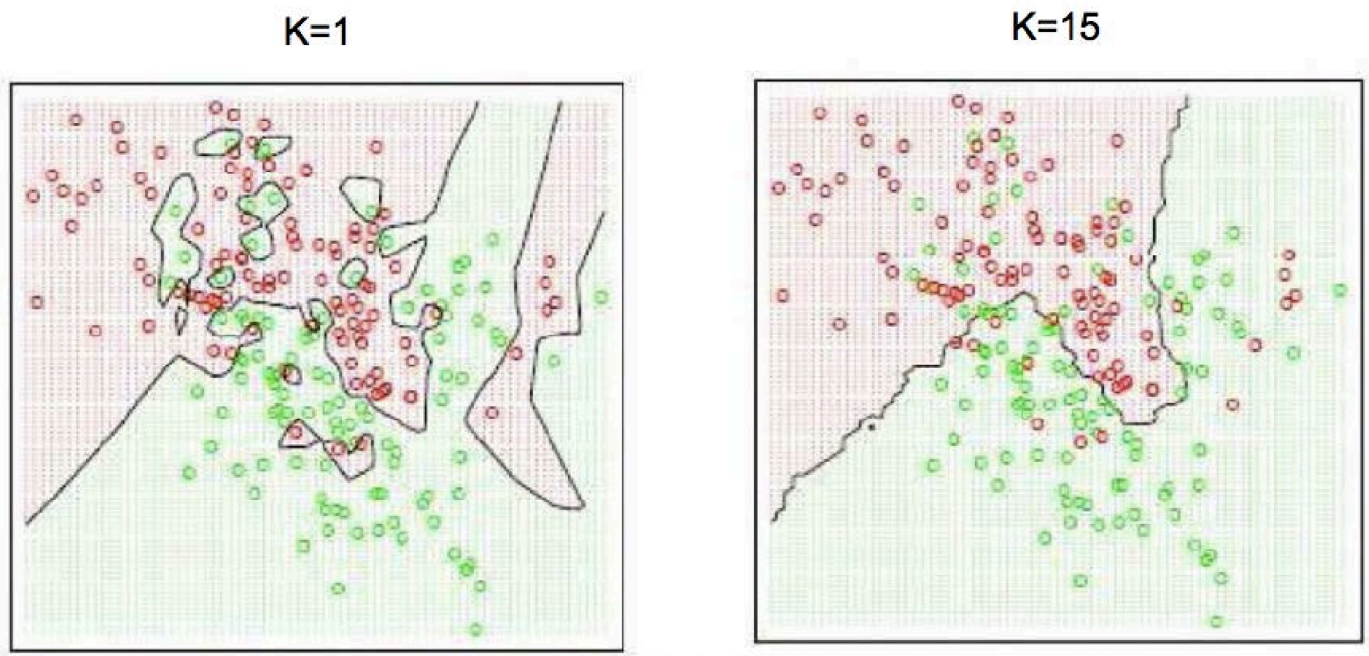
Two example distances: 100->011 has distance 3 (red path); 010->111 has distance 2 (blue path)

- Mahalanobis distance
  - based on correlations between variables by which different patterns can be identified and analyzed

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1} (x-\mu)}$$

S is the covariance matrix

EFFECT OF K



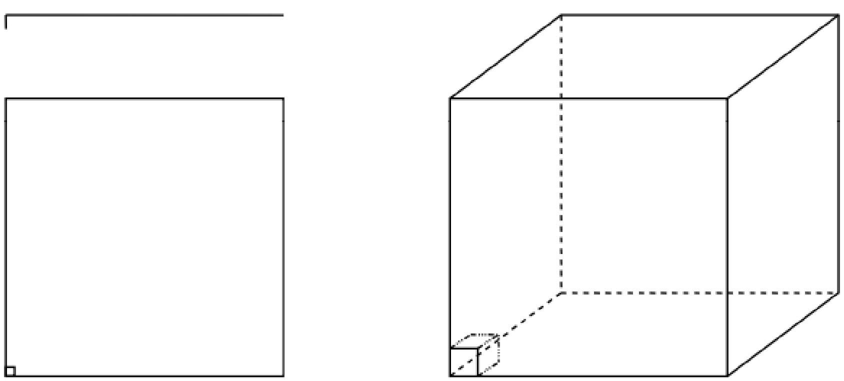
Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

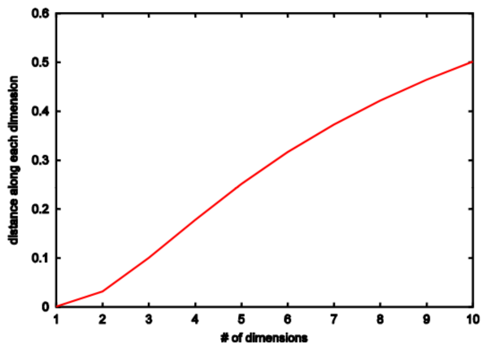
Larger k produces smoother boundary effect and can reduce the impact of class label noise.

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

CURSE OF DIMENSIONALITY IN K-NN

- kNN breaks down in high-dimensional space
  - “Neighborhood” becomes very large.
- Assume 5000 points uniformly distributed in the unit hypercube 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 to get a square that contains 0.001 of the volume.
  - In d dimensions, we must go (0.001) 1/d



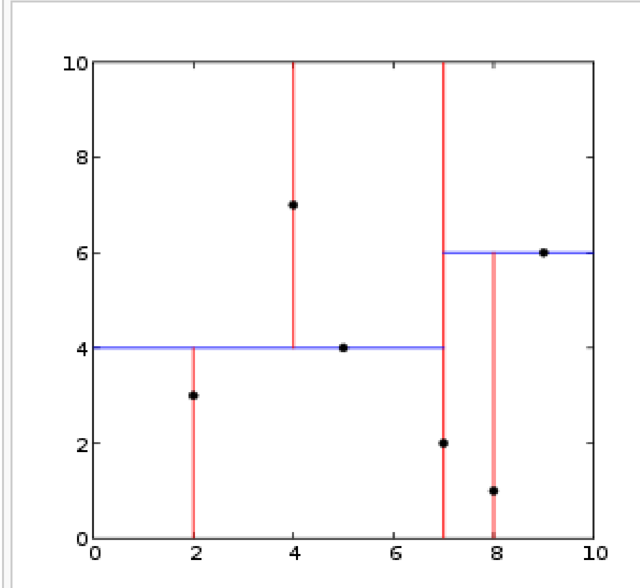


FINDING NEAREST NEIGHBORS WITH K-D TREES

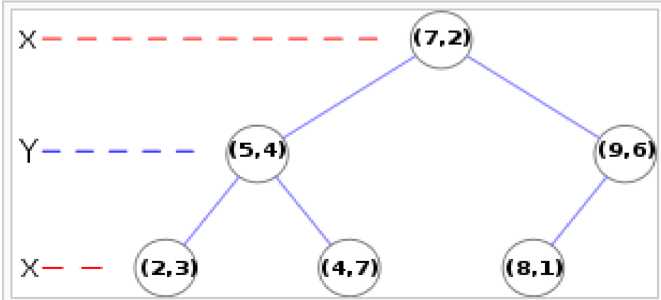
- Def: A balanced binary tree over data wit an arbitrary number of dimensions.
- Construction
  - Start with set of examples
  - At root, split examples along the *i*th dim (*i* mode *n*) by testing

    $x_i < m$. (half in right, half in left)

    (let m be median of examples at ith dim)
  - Recursively make a tree from the left and right set of examples
  - stopping when there are fewer than two examples left.
- To choose a dimension to split on at each node of the tree,
  - select dimension i mod n at level i of the tree.
  - split on the dimension that has the widest spread of values.
- Note: may need to split on any given dimension several times as \proceed down the tree



k-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2).



The resulting *k*-d tree.

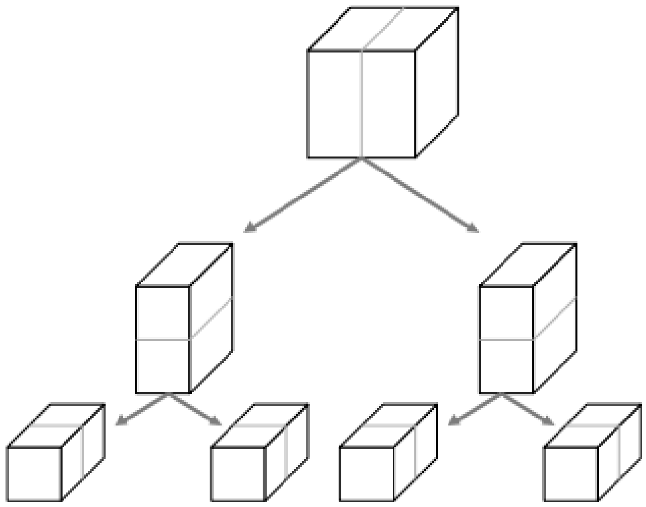
http://en.wikipedia.org/wiki/K-d_tree

FINDING NEAREST NEIGHBORS WITH K-D TREES CONT.

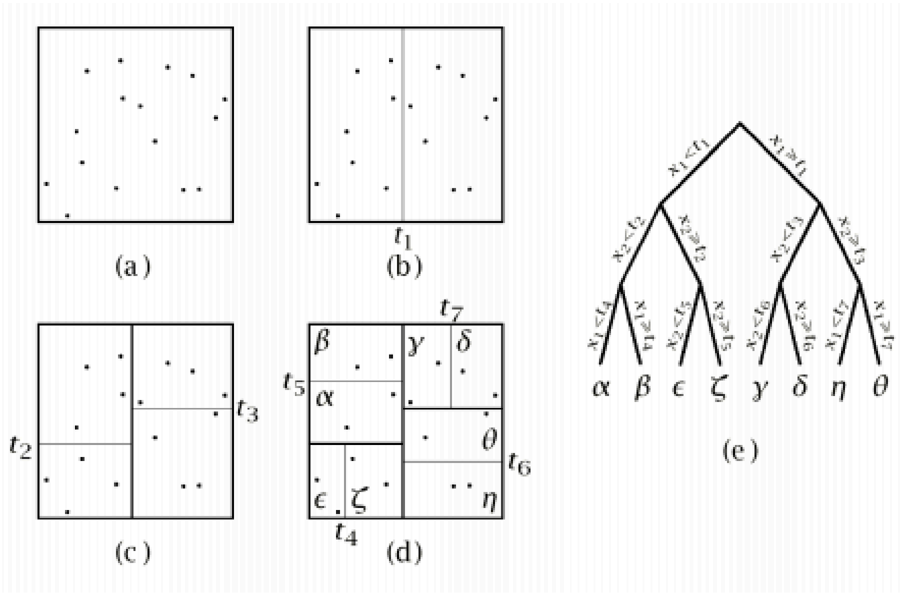
- Exact lookup from a k-d tree is just like lookup from a binary tree, but nearest neighbor lookup is more complicated.
- NN Searching:
  - Search down the branches, splitting the examples in half
  - Discard half of the example are discarded if query is far from the boundary
  - If query falls very close to the dividing boundary:
    - The query point itself might be on the left hand side of the boundary, but one or more of the k nearest neighbors might actually be on the right-hand side.
    - Test by computing the distance of the query point to the dividing boundary, and then searching both sides if we can't find k examples on the left that are closer than this distance.
- Because of boundary checking problem, k-d trees are appropriate only when there are many more examples than dimensions
  - Preferably at least 2^n examples.
  - k-d trees work well with up to 10 dimensions with thousands of examples or up to 20 dimensions with millions of examples.
  - If we don't have enough examples, lookup is no faster than a linear scan of the entire data set.

k-d tree example

Data structure (3D case)



Partitioning (2D case)



LOCALITY-SENSITIVE HASHING

- Hash codes rely on an *exact* match, how can we use hashing scheme for NN problem?
- Hashes cannot solve NN(k, xq) exactly, but with a clever use of randomized algorithms, we can find an approximate solution
- IDEA: Place near points grouped together in the same bin
- **Approximate near-neighbor problem:**
  - If there is a point x_j that is within a radius r of x_q, than with high probability the algorithm will find a point x_j' that is within distance $c*r$ of q.
  - If there is no point within radius r then the algorithm is allowed to report failure.
  - The values of c and "high probability" are parameters of the algorithm.

LOCALITY-SENSITIVE HASHING CONT.

- Define the **hash function** g(x) such that,
  - for any two points x_j and x_j', the probability that they have the same hash code is small if their distance is more that c*r and is high if their distance is less than r.
- **Idea:** if two points are close together in an n-D space, then they will necessarily be close when projected down onto a 1D space(a line).
  - We can discretize the line into bins (hash buckets) so that, with high probability, near points project down to exactly the same bin.
  - Thus, the bin for point xq contains many (but not all) points that are near to xq, as well as some points that are far away

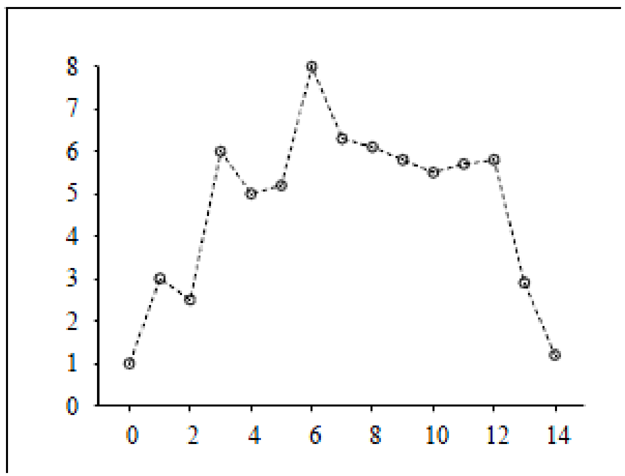
LOCALITY-SENSITIVE HASHING CONT.

- Idea of **locality-sensitive hash** (LSH): create multiple random projections and combine them.
- Creating LSH
  - Choose l different random projections (l subset of bit-string representation)
  - Create l hash tables, $g_1(x)$, $g_2(x)$, …,$g_l(x)$
  - Enter all examples to hash table.
- Searching LSH
  - For query x_q , fetch the set of point in bin $g_k(x)$ for each k.
  - Union these sets together into a set of candidate point, C.
  - Compute actual distance x_q for points in C.

NONPARAMETRIC REGRESSION

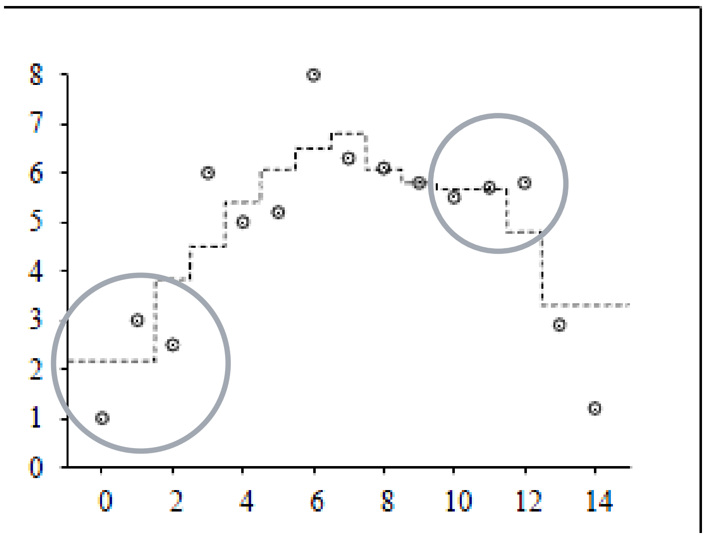
Piecewise linear nonparametric regression (Connect the dot):

- Creates a function **h(x)** that, when given a query **xq**, solves the ordinary linear regression problem with just two points: the training examples immediately to the left and right of **xq**.
- When noise is low performance not too bad, which is why it is a standard feature of charting in spreadsheets
- But when the data are noisy, the resulting function is spiky, and does not generalize well



NONPARAMETRIC REGRESSION CONT.

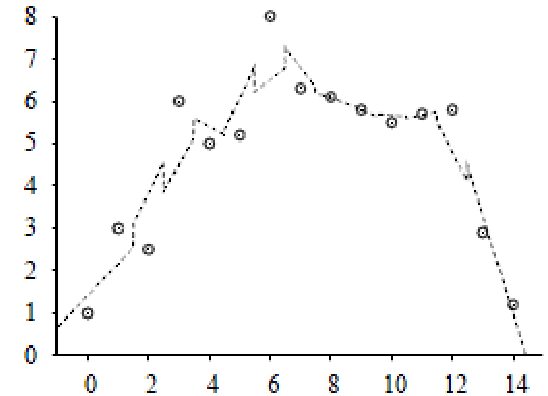
- k-**nearest-neighbors regression**
  - improves on connect-the-dots using k nearest neighbors (here 3) instead of using just the two examples to the left and right of a query point **xq**
  - h(x) is the mean value of the k points,Sum(yj/k).
  - A larger value of **k** tends to smooth out the magnitude of the spikes, although the resulting function has discontinuities.



Notice that at the outlying points, near x=0 and x=14, the estimates are poor because all the evidence comes from one side (the interior)

NONPARAMETRIC REGRESSION CONT.

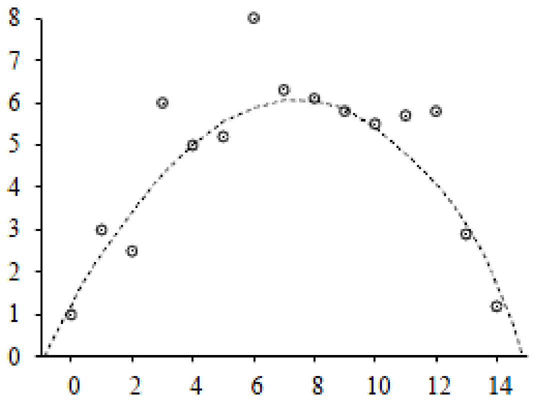
- k-nearest-neighbor linear regression
  - finds the best line through the k examples
  - This does a better job of capturing trends at the outliers, but is still discontinuous.



(c)

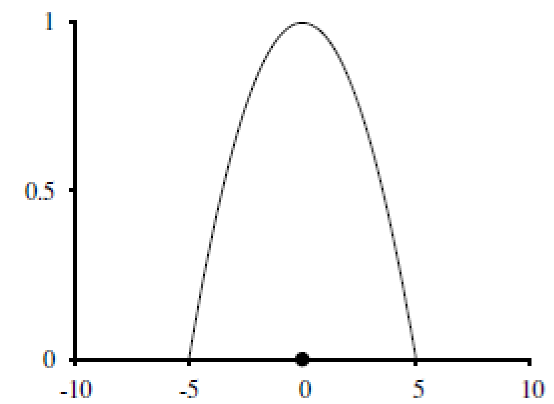
NONPARAMETRIC REGRESSION CONT.

- **Locally weighted regression**
- gives us the advantages of nearest neighbors, without the discontinuities.
- Idea: weight the examples
  - For each query point **xq**, the examples that are close to **xq** are weighted heavily, and the examples that are farther away are weighted less heavily or not at all.
  - The decrease in weight over distance is always gradual, not sudden.



NONPARAMETRIC REGRESSION CONT.

- Use **kernel function** to decide how much to weight each example.
  - *K(Distance(xj, xq))*, where xq is a query point that is a given distance from xj
  - *K* should be symmetric around 0 and have a maximum at 0.
  - The area under the kernel must remain bounded as we go to ±INF.
  - latest research suggests that the choice of kernel shape doesn't matter much but do have to be careful about the width of the kernel.



quadratic kernel, $\mathcal{K}(d) = \max(0, 1 - (2|x|/k)^2)$,

NONPARAMETRIC REGRESSION CONT.

- After selecting the kernel, regression follow:
- For a given query point **xq**, solve the following weighted regression problem using gradient descent:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j \mathcal{K}(Distance(\mathbf{x}_q, \mathbf{x}_j))\,(y_j - \mathbf{w} \cdot \mathbf{x}_j)^2 \,,$$

- where Distance is any of the distance metrics discussed for nearest neighbors
- The answer is

$$h(\mathbf{x}_q) = \mathbf{w}^* \cdot \mathbf{x}_q .$$

- Note: we need to solve a new regression problem for *every* query point