CSC 411: Lecture 05: Nearest Neighbors

Raquel Urtasun & Rich Zemel

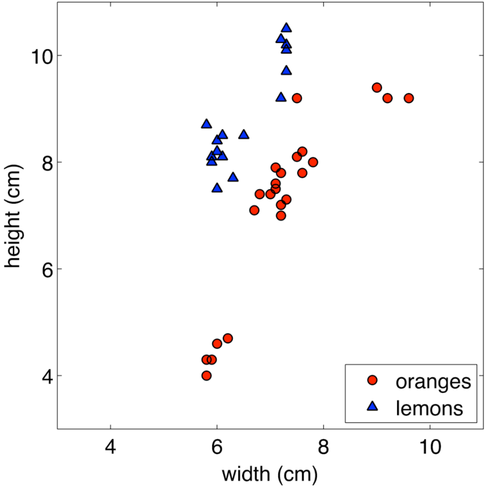
University of Toronto

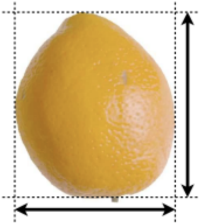
Sep 28, 2015

Today

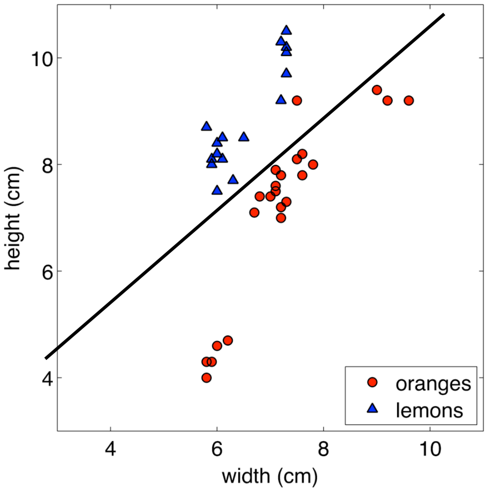
- Non-parametric models
  - distance
  - non-linear decision boundaries

Classification: Oranges and Lemons



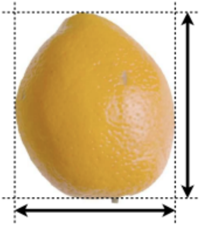


Classification: Oranges and Lemons



Can construct simple linear decision boundary:

$y = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$



What is the meaning of "linear" classification

- Classification is intrinsically non-linear
  - It puts non-identical things in the same class, so a difference in the input vector sometimes causes zero change in the answer
- Linear classification means that the part that adapts is linear (just like linear regression)

$$z(x) = \mathbf{w}^T \mathbf{x} + w_0$$

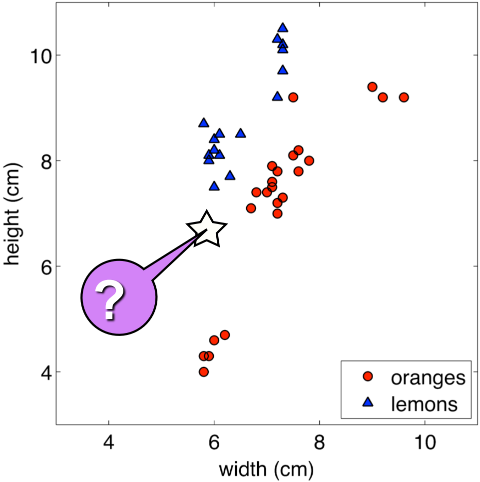
  with adaptive $\mathbf{w}$, w_0

- The adaptive part is follow by a non-linearity to make the decision

$$y(\mathbf{x}) = f(z(\mathbf{x}))$$

- What f have we seen so far in class?

Classification as Induction



Instance-based Learning

- Alternative to parametric model is non-parametric
- Simple methods for approximating discrete-valued or real-valued target functions (classification or regression problems)
- Learning amounts to simply storing training data
- Test instances classified using similar training instances
- Embodies often sensible underlying assumptions:
  - Output varies smoothly with input
  - Data occupies sub-space of high-dimensional input space

Nearest Neighbors

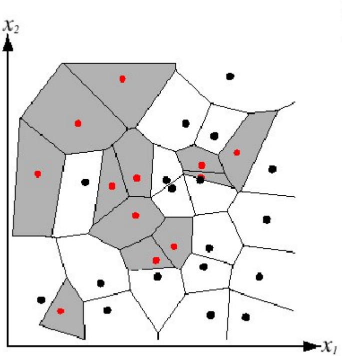
- Assume training examples correspond to points in d-dimensional Euclidean space
- Target function value for new query estimated from known value of nearest training example(s)
- Distance typically defined to be Euclidean:

$$||\mathbf{x}^{(a)} - \mathbf{x}^{(b)}||_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

- Algorithm
  1. find example $(\mathbf{x}^*, t^*)$ closest to the test instance $\mathbf{x}^{(q)}$
  2. output $y^{(q)} = t^*$
- Note: we don't need to compute the square root. Why?

Nearest Neighbors Decision Boundaries

- Nearest neighbor algorithm does not explicitly compute decision boundaries, but these can be inferred
- Decision boundaries: Voronoi diagram visualization
  - show how input space divided into classes
  - each line segment is equidistant between two points of opposite classes



k Nearest Neighbors

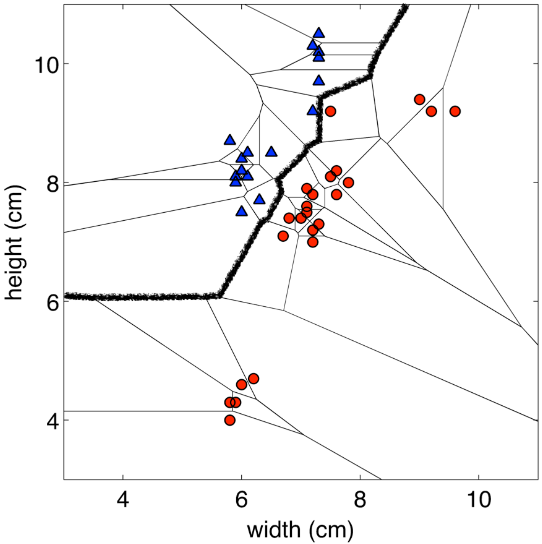
- Nearest neighbors sensitive to mis-labeled data ("class noise") $\rightarrow$ smooth by having k nearest neighbors vote
- Algorithm:
  1. find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
  2. classification output is majority class

$$y = arg \max_{t^{(z)}} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$$

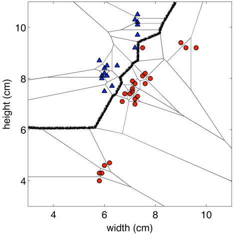
k Nearest Neighbors: Issues & Remedies

- Some attributes have larger ranges, so are treated as more important
  - normalize scale
- Irrelevant, correlated attributes add noise to distance measure
  - eliminate some attributes
  - or vary and possibly adapt weight of attributes
- Non-metric attributes (symbols)
  - Hamming distance
- Brute-force approach: calculate Euclidean distance to test point from each stored point, keep closest: $O(dn^2)$. We need to reduce computational burden:
  1. Use subset of dimensions
  2. Use subset of examples
     - Remove examples that lie within Voronoi region
     - Form efficient search tree (kd-tree), use Hashing (LSH), etc

Decision Boundary K-NN



K-NN Summary



- Single parameter (k) → how do we set it?
- Naturally forms complex decision boundaries; adapts to data density
- Problems:
  - Sensitive to class noise.
  - Sensitive to dimensional scales.
  - Distances are less meaningful in high dimensions
  - Scales with number of examples
- Inductive Bias: What kind of decision boundaries do we expect to find?