# CSC 411: Lecture 05: Nearest Neighbors

Rich Zemel, Raquel Urtasun and Sanja Fidler

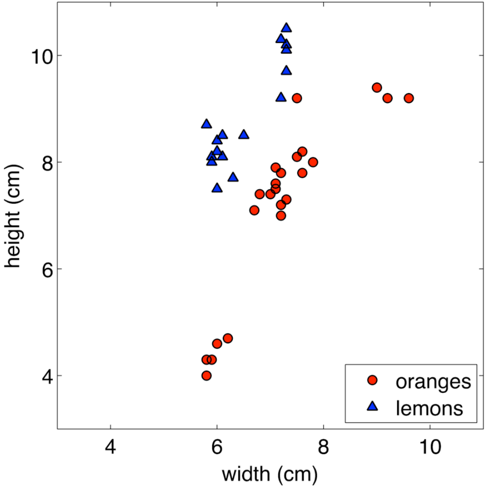
University of Toronto

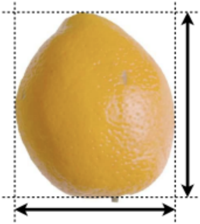
# Today

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

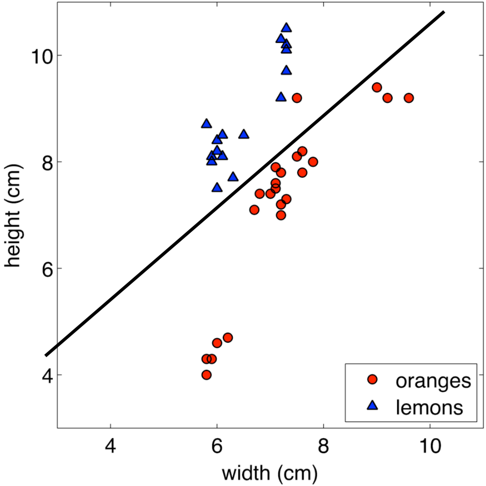
Note: We will mainly use today's method for classification, but it can also be used for regression

# Classification: Oranges and Lemons



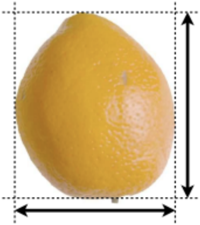



# Classification: Oranges and Lemons




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

# 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 $\mathbf{w}$, $w_0$

# 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 followed by a non-linearity to make the decision

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

# 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)

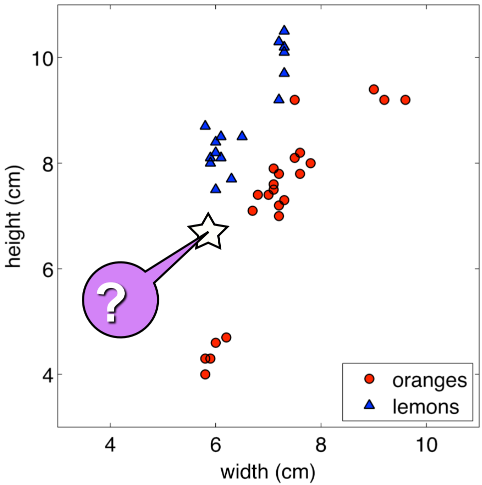
$$z(x) = \mathbf{w}^T\mathbf{x} + w_0$$

  with adaptive $\mathbf{w}, w_0$
- The adaptive part is followed by a non-linearity to make the decision

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

- What functions $f()$ have we seen so far in class?

# Classification as Induction



# Instance-based Learning

- Alternative to parametric models are non-parametric models

# Instance-based Learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)

# Instance-based Learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- Learning amounts to simply storing training data

# Instance-based Learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- Learning amounts to simply storing training data
- Test instances classified using similar training instances

# Instance-based Learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for 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

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$

# Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the nearest training example(s)

# Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the nearest training example(s)
- Distance typically defined to be Euclidean:

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

# Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the 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 $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to the test instance $\mathbf{x}$. That is:

$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\text{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

# Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the 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 $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to the test instance $\mathbf{x}$. That is:

$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\text{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

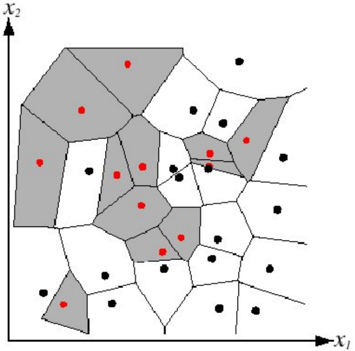
- Note: we don't really need to compute the square root. Why?

# Nearest Neighbors: Decision Boundaries

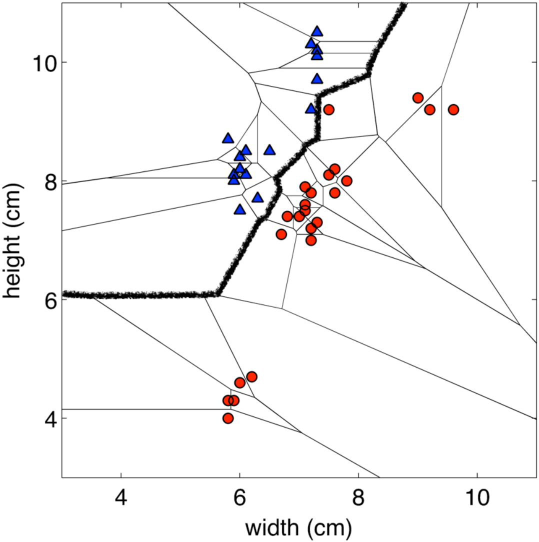
- Nearest neighbor algorithm does not explicitly compute decision boundaries, but these can be inferred

# 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

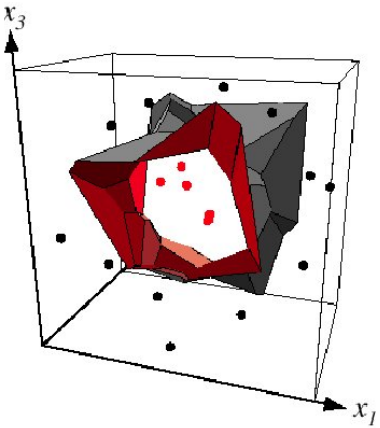


# Nearest Neighbors: Decision Boundaries




Example: 2D decision boundary

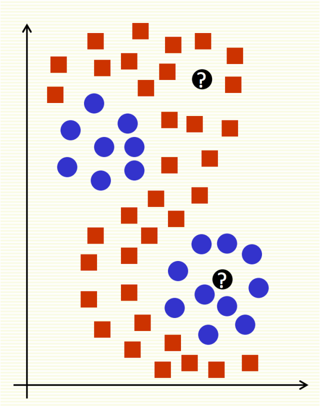
# Nearest Neighbors: Decision Boundaries




Example: 3D decision boundary

# Nearest Neighbors: Multi-modal Data

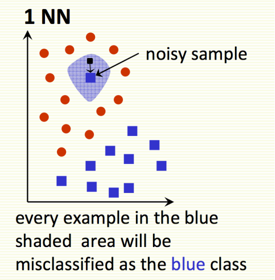
- Nearest Neighbor approaches can work with multi-modal data




[Slide credit: O. Veksler]

# Nearest Neighbors

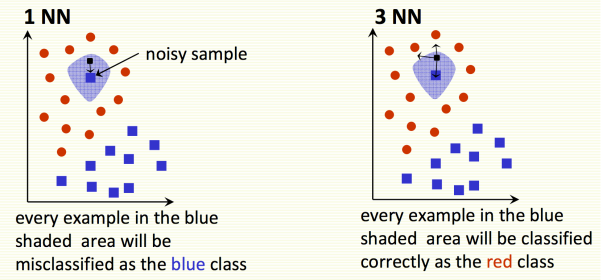
[Pic by Olga Veksler]




- Nearest neighbors sensitive to mis-labeled data ("class noise"). Solution?

# k-Nearest Neighbors

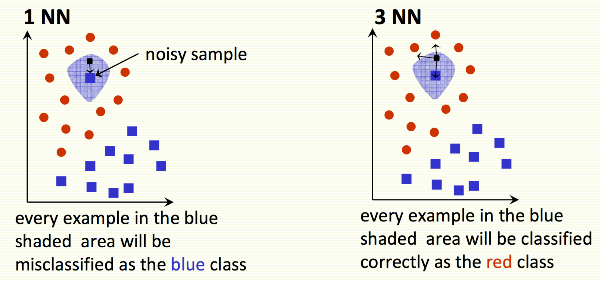
[Pic by Olga Veksler]




- Nearest neighbors sensitive to mis-labeled data ("class noise"). Solution?
- Smooth by having k nearest neighbors vote

# k-Nearest Neighbors

[Pic by Olga Veksler]




- Nearest neighbors sensitive to mis-labeled data ("class noise"). Solution?
- Smooth by having k nearest neighbors vote

**Algorithm (kNN)**:

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

How do we choose $k$?

- Larger $k$ may lead to better performance
- But if we set $k$ too large we may end up looking at samples that are not neighbors (are far away from the query)

# k-Nearest Neighbors

How do we choose $k$?

- Larger $k$ may lead to better performance
- But if we set $k$ too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use cross-validation to find $k$

# k-Nearest Neighbors

How do we choose $k$?

- Larger $k$ may lead to better performance
- But if we set $k$ too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use cross-validation to find $k$
- Rule of thumb is $k < sqrt(n)$, where $n$ is the number of training examples

[Slide credit: O. Veksler]

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters
- Irrelevant, correlated attributes add noise to distance measure

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters
- Irrelevant, correlated attributes add noise to distance measure
  - eliminate some attributes

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters
- Irrelevant, correlated attributes add noise to distance measure
  - eliminate some attributes
  - or vary and possibly adapt weight of attributes

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters
- Irrelevant, correlated attributes add noise to distance measure
  - eliminate some attributes
  - or vary and possibly adapt weight of attributes
- Non-metric attributes (symbols)

# k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of $\mathbf{x}$) have larger ranges, they are treated as more important
  - normalize scale
    - Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
    - Linearly scale each dimension to have 0 mean and variance 1 (compute mean $\mu$ and variance $\sigma^2$ for an attribute $x_j$ and scale: $(x_j - m)/\sigma$)
  - be careful: sometimes scale matters
- 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

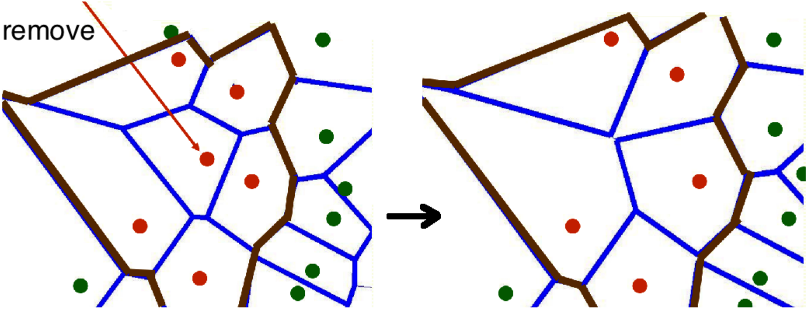
# k-Nearest Neighbors: Issues (Complexity) & Remedies

- Expensive at test time: To find one nearest neighbor of a query point **x**, we must compute the distance to all N training examples. Complexity: $O(kdN)$ for kNN
  - Use subset of dimensions
  - Pre-sort training examples into fast data structures (e.g., kd-trees)
  - Compute only an approximate distance (e.g., LSH)
  - Remove redundant data (e.g., condensing)
- Storage Requirements: Must store all training data
  - Remove redundant data (e.g., condensing)
  - Pre-sorting often increases the storage requirements
- High Dimensional Data: "Curse of Dimensionality"
  - Required amount of training data increases exponentially with dimension
  - Computational cost also increases

[Slide credit: David Claus]

# k-Nearest Neighbors Remedies: Remove Redundancy

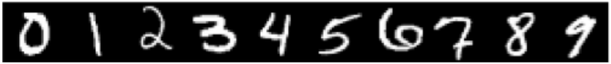
- If all Voronoi neighbors have the same class, a sample is useless, remove it




[Slide credit: O. Veksler]

# Example: Digit Classification

- Decent performance when lots of data



- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

| | Test Error Rate (%) |
|---|---|
| Linear classifier (1-layer NN) | 12.0 |
| K-nearest-neighbors, Euclidean | 5.0 |
| K-nearest-neighbors, Euclidean, deskewed | 2.4 |
| K-NN, Tangent Distance, 16x16 | 1.1 |
| K-NN, shape context matching | 0.67 |
| 1000 RBF + linear classifier | 3.6 |
| SVM deg 4 polynomial | 1.1 |
| 2-layer NN, 300 hidden units | 4.7 |
| 2-layer NN, 300 HU, [deskewing] | 1.6 |
| LeNet-5, [distortions] | 0.8 |
| Boosted LeNet-4, [distortions] | 0.7 |

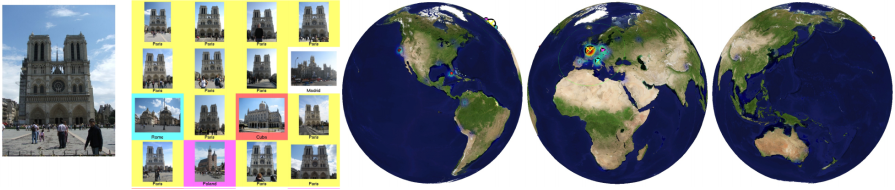
# Fun Example: Where on Earth is this Photo From?

- Problem: Where (e.g., which country or GPS location) was this picture taken?

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

# Fun Example: Where on Earth is this Photo From?

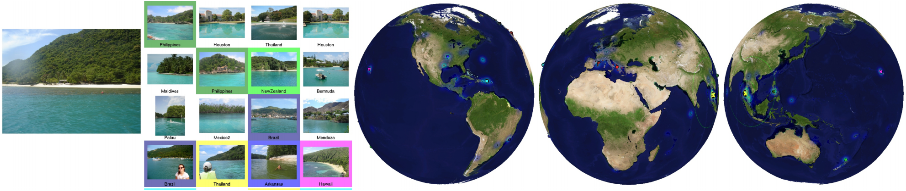
- Problem: Where (e.g., which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with GPs info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN!




[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

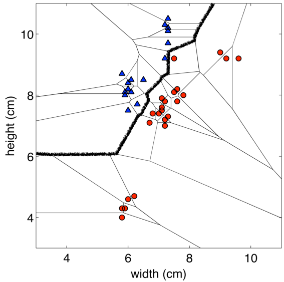
# Fun Example: Where on Earth is this Photo From?

- Problem: Where (eg, which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with gps info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN (large $k$ better, they use $k = 120$)!



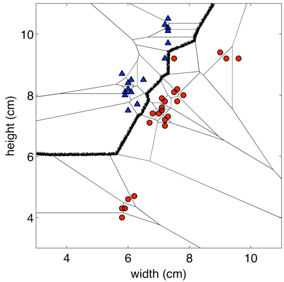

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

# K-NN Summary



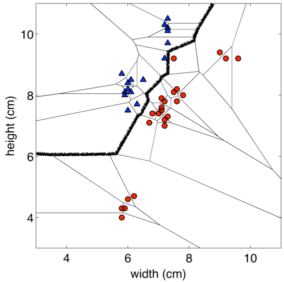

- Naturally forms complex decision boundaries; adapts to data density

# K-NN Summary



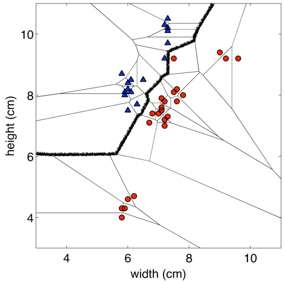

- Naturally forms complex decision boundaries; adapts to data density
- If we have lots of samples, kNN typically works well

# K-NN Summary




- Naturally forms complex decision boundaries; adapts to data density
- If we have lots of samples, kNN typically works well
- Problems:
  - Sensitive to class noise
  - Sensitive to scales of attributes
  - Distances are less meaningful in high dimensions
  - Scales linearly with number of examples

# K-NN Summary




- Naturally forms complex decision boundaries; adapts to data density
- If we have lots of samples, kNN typically works well
- Problems:
  - Sensitive to class noise
  - Sensitive to scales of attributes
  - Distances are less meaningful in high dimensions
  - Scales linearly with number of examples
- Inductive Bias: What kind of decision boundaries do we expect to find?