10-301/601: Introduction to Machine Learning Lecture 4 –KNNs

Henry Chai

5/22/23

Front Matter

- Announcements:
  - PA1 released 5/18, due 5/25 at 11:59 PM
  - Quiz 1: Decision Trees on 5/19 (tomorrow!)
    - The quiz will begin promptly at 11 AM, please show up on time!
    - Closed book, closed notes, no calculators.
- Recommended Readings:
  - Daumé III, Chapter 3: Geometry and Nearest Neighbors

Real-valued Features

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

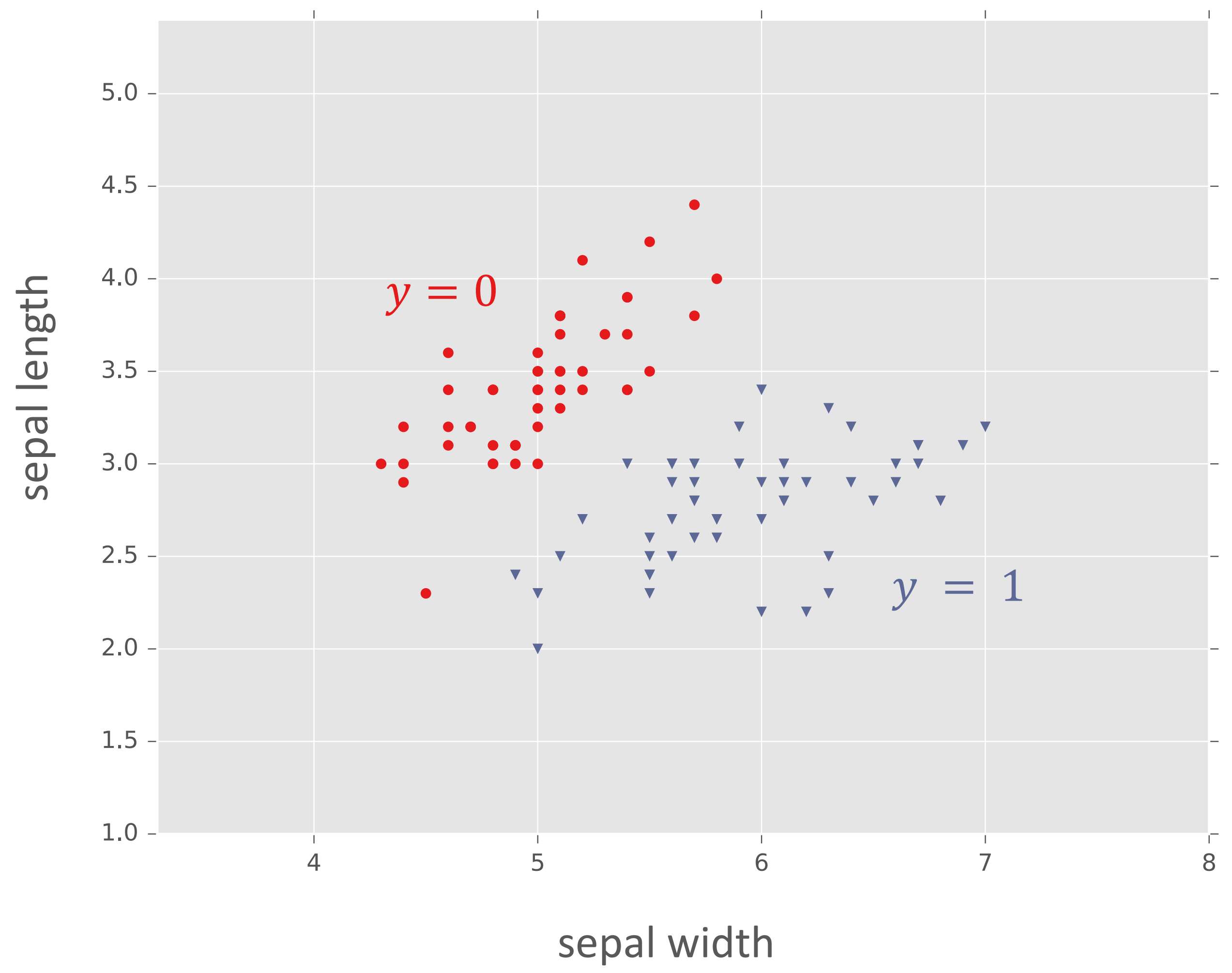
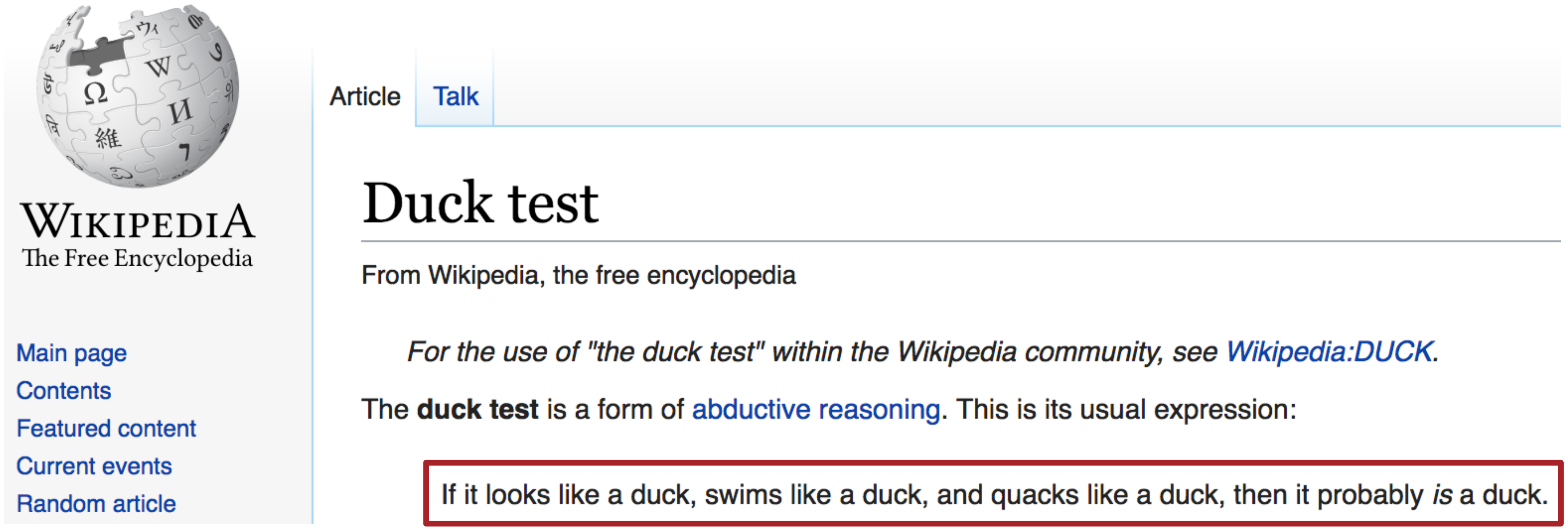


Figure courtesy of Matt Gormley



Duck test

From Wikipedia, the free encyclopedia

For the use of "the duck test" within the Wikipedia community, see Wikipedia:DUCK.

The **duck test** is a form of abductive reasoning. This is its usual expression:

If it looks like a duck, swims like a duck, and quacks like a duck, then it probably *is* a duck.

The Duck Test

Source: https://en.wikipedia.org/wiki/Duck_test

The Duck Test for Machine Learning

- Classify a point as the label of the "most similar" training point
- Idea: given real-valued features, we can use a distance metric to determine how similar two data points are
- A common choice is Euclidean distance:

$$d(x, x') = \|x - x'\|_2 = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2}$$

- An alternative is the Manhattan distance:

$$d(x, x') = \|x - x'\|_1 = \sum_{d=1}^{D} |x_d - x'_d|$$

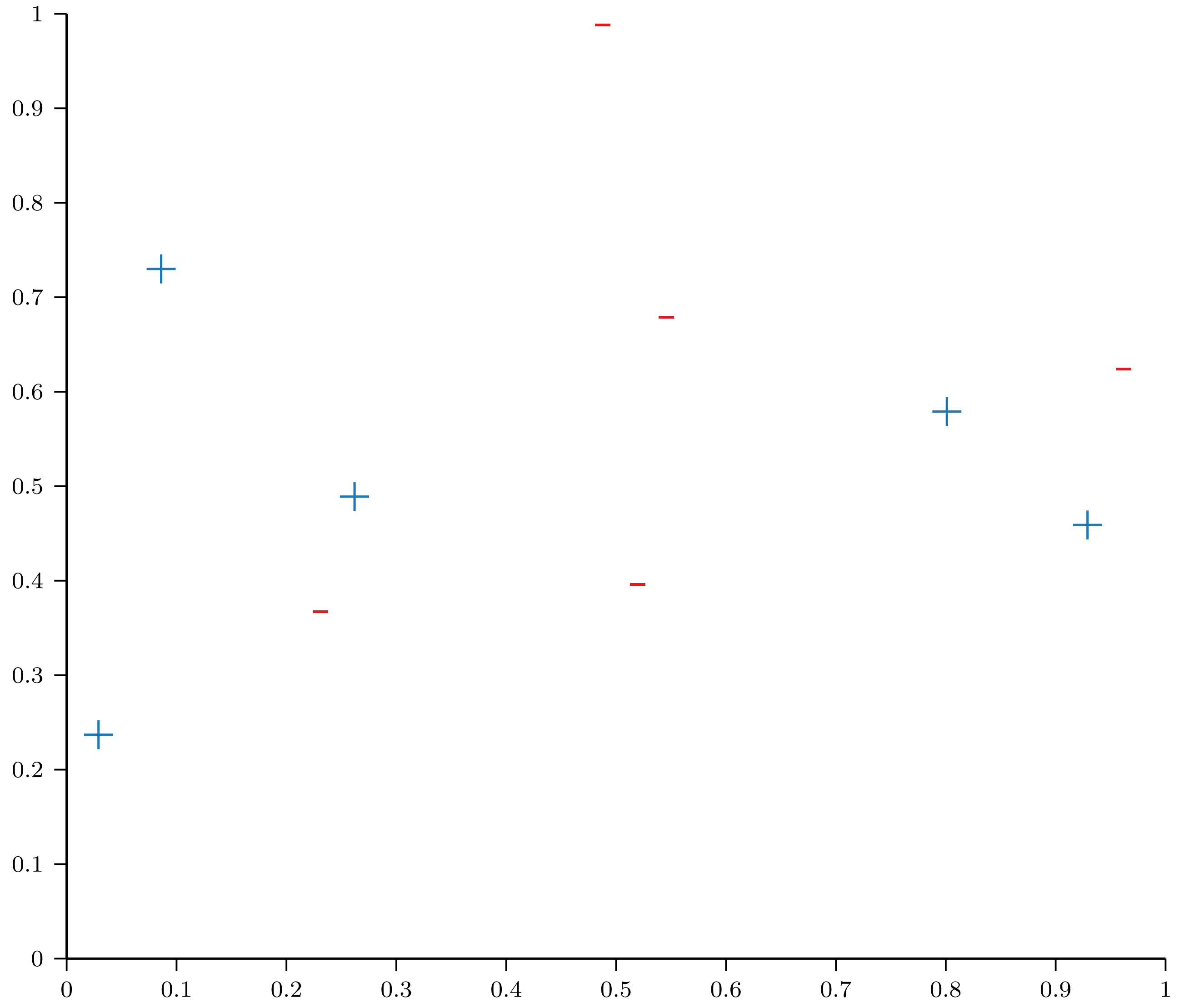
$$L \quad p\text{-norm} = \sqrt[p]{\sum_{d=1}^{D} |(x_d - x'_d)|^p}$$

Nearest Neighbor: Pseudocode

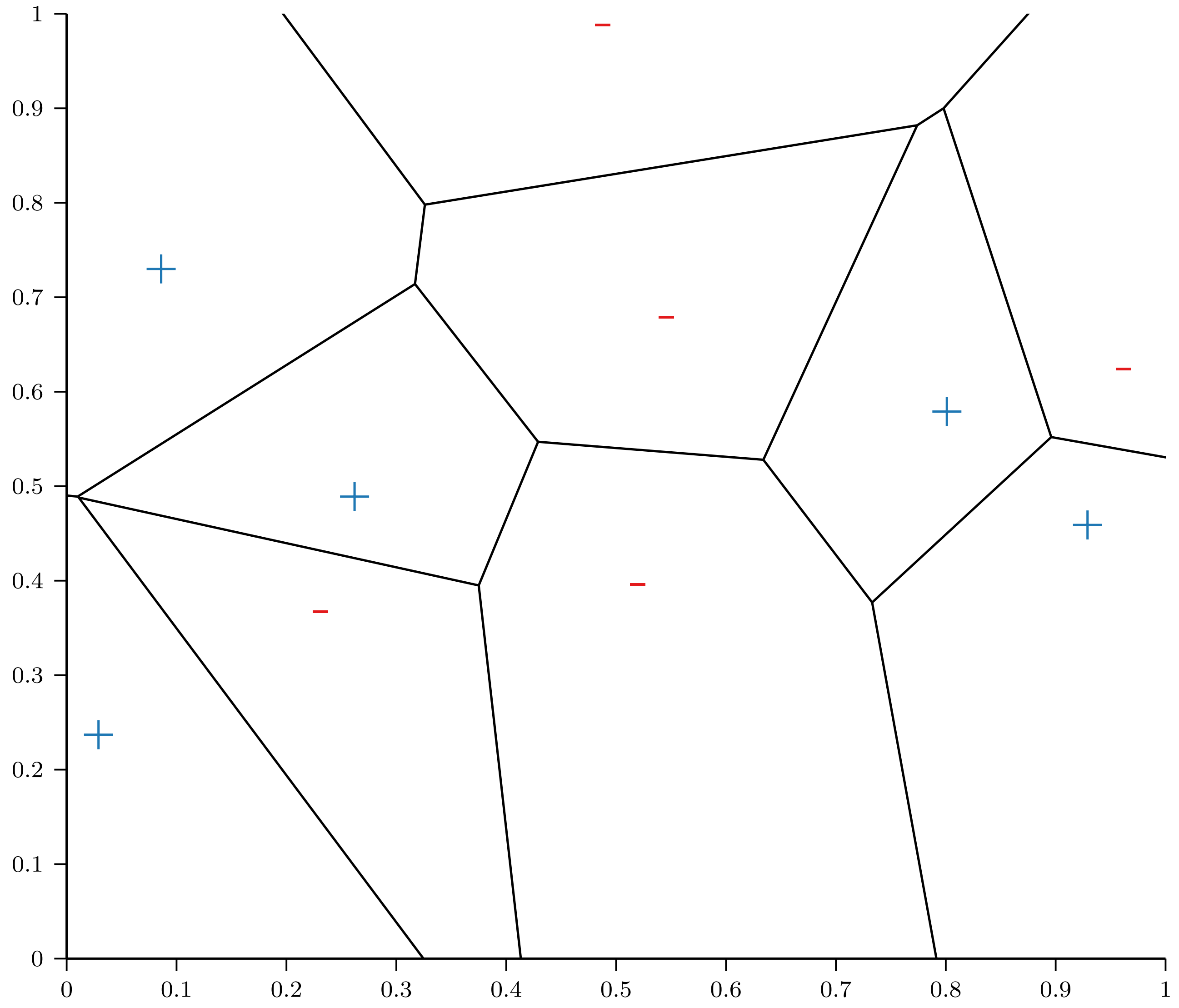
```
def train($D_{train}$):
    store $D_{train}$

def predict($x'$):
    Find the nearest neighbor to $x'$
    in $D_{train}$, $x^{(i)}$, according to
    distance metric $d$
    return $y^{(i)}$
```

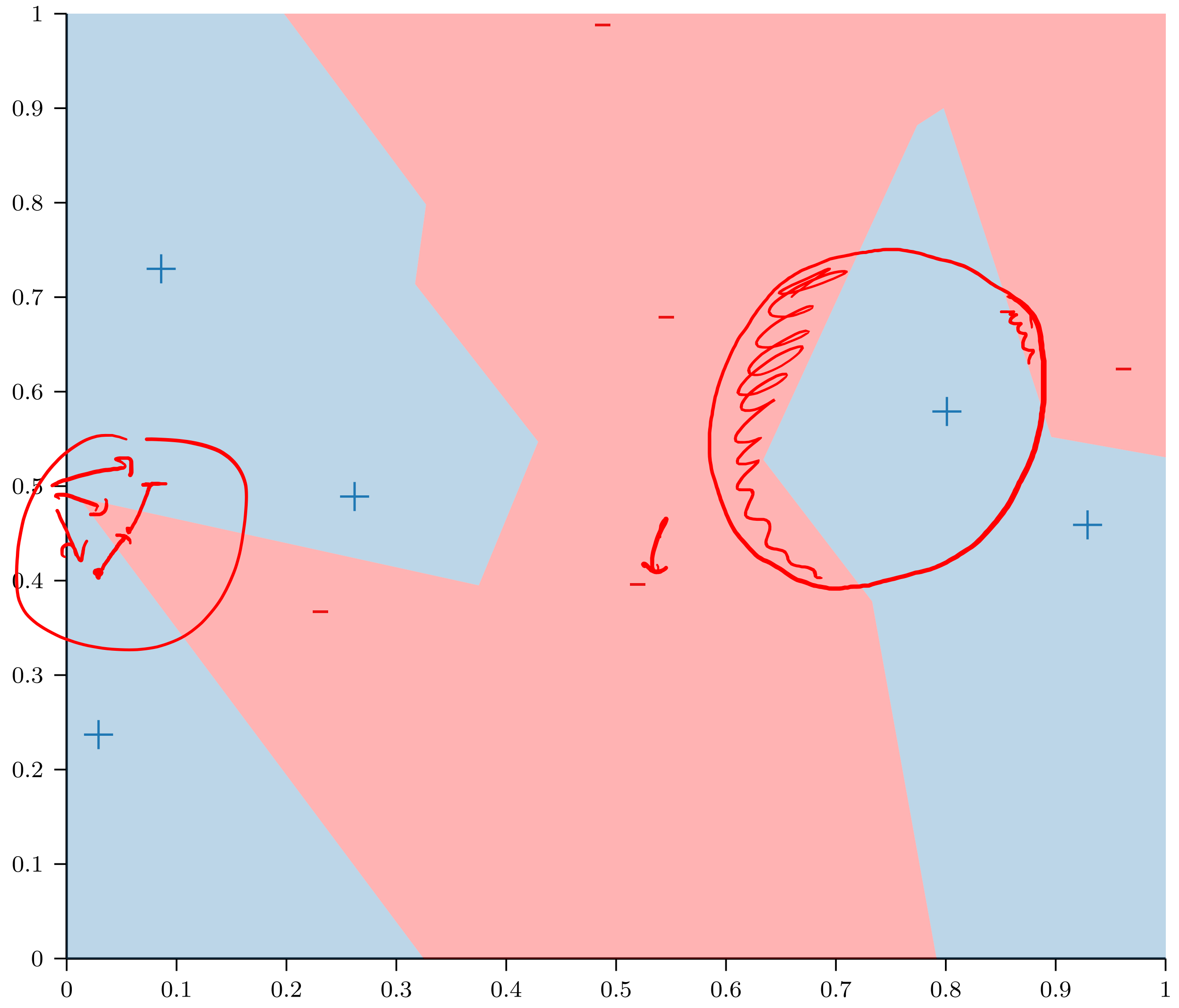
Nearest Neighbor: Example



Nearest Neighbor: Example



Nearest Neighbor: Example



The Nearest Neighbor Model

- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***

⋮

- Always has zero training error…

Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 \ast$ the Bayes error rate (the optimal classifier)
- Interpretation: “In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor.”

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964

But why limit ourselves to just one neighbor?

- Claim: under certain conditions, as $n \to \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2\ *$ the Bayes error rate (the optimal classifier)
- Interpretation: "In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964

k-Nearest Neighbors (kNN)

- Classify a point as the most common label among the labels of the k nearest training points
- Tie-breaking (in case of even k and/or more than 2 classes)

- add a neighbor
- remove a neighbor
- weight votes by distance
- draw a circle around the data point w/ compute shaded areas
- return randomly or weighted randomly
- majority vote over all data points
- change d

k-Nearest Neighbors (kNN): Pseudocode

```
def train($D_{train}$):
    store $D_{train}$
def predict($x'$)
    return majority_vote(labels of the
        k nearest neighbors to
        $x'$ in $D_{train}$ according
        to distance metric d)
```

kNN on Fisher Iris Data

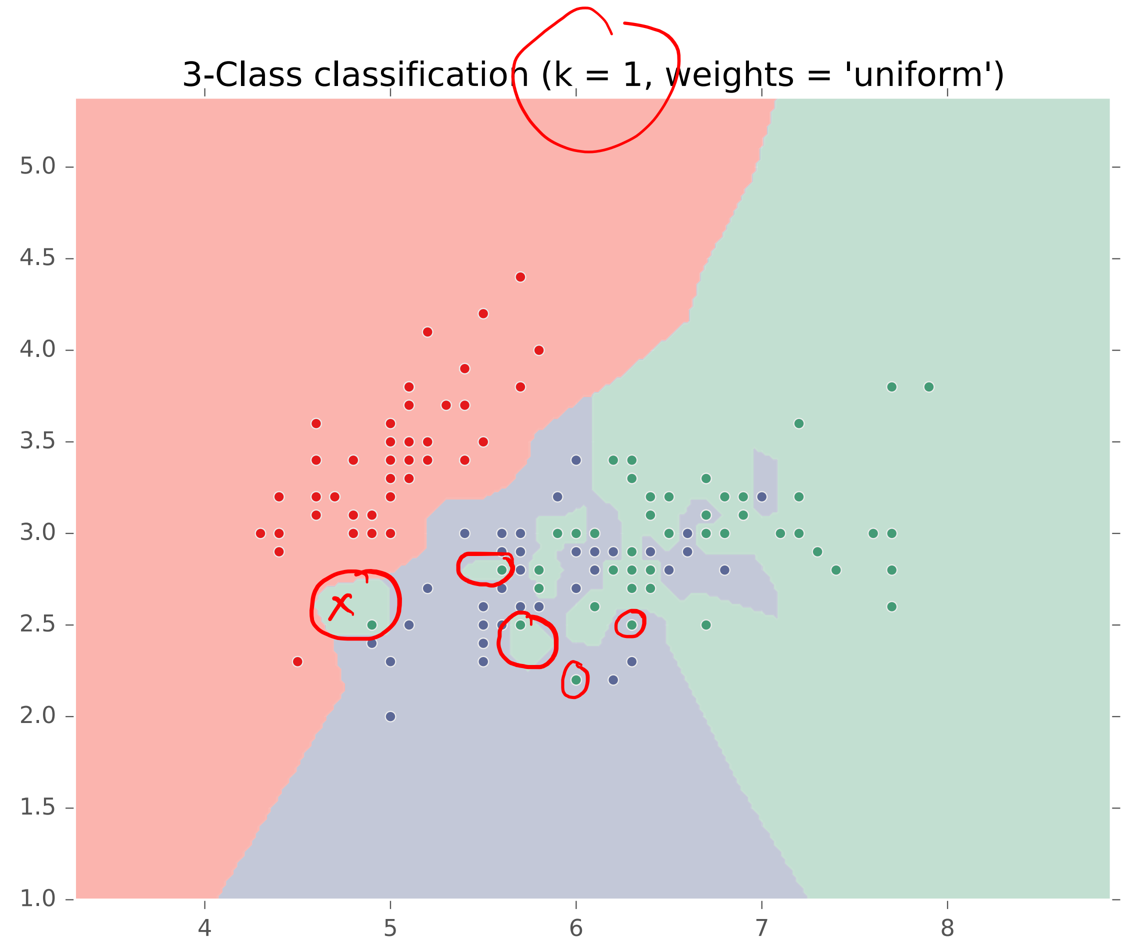


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

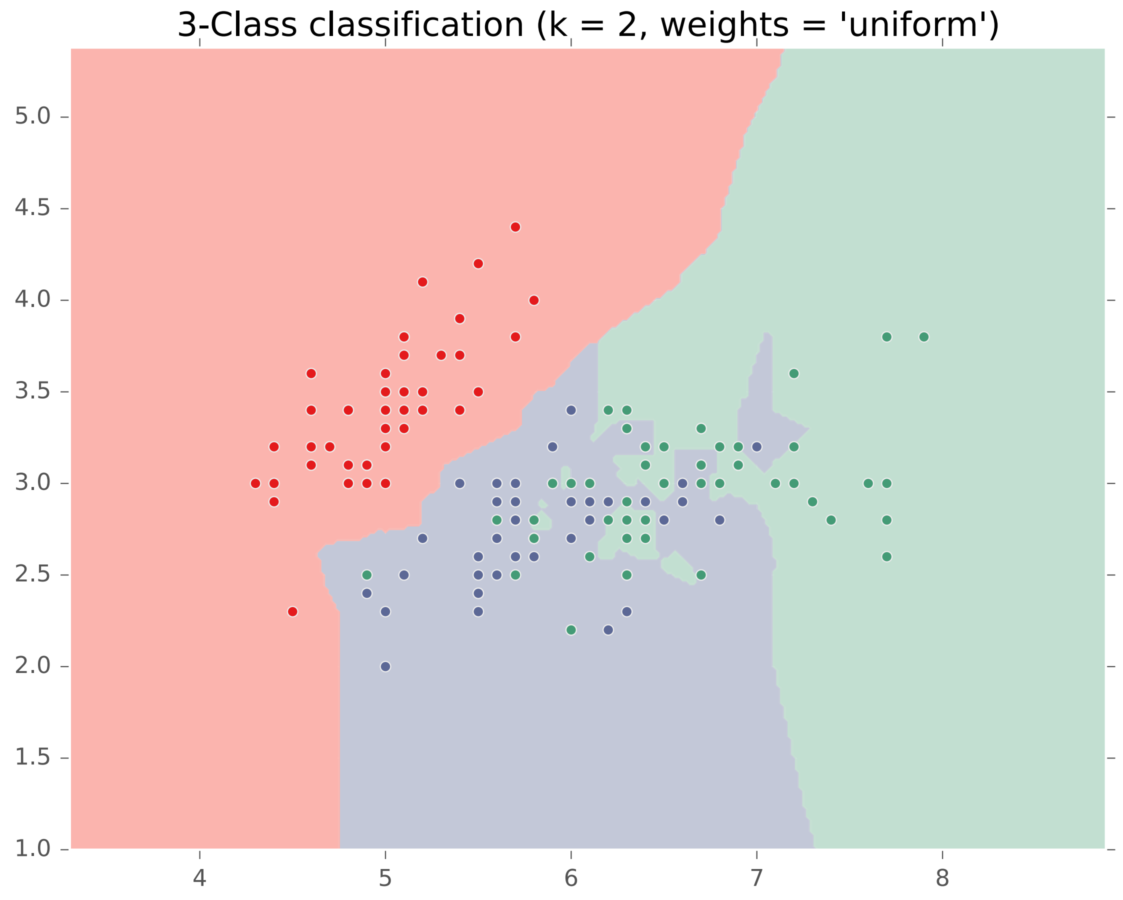


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

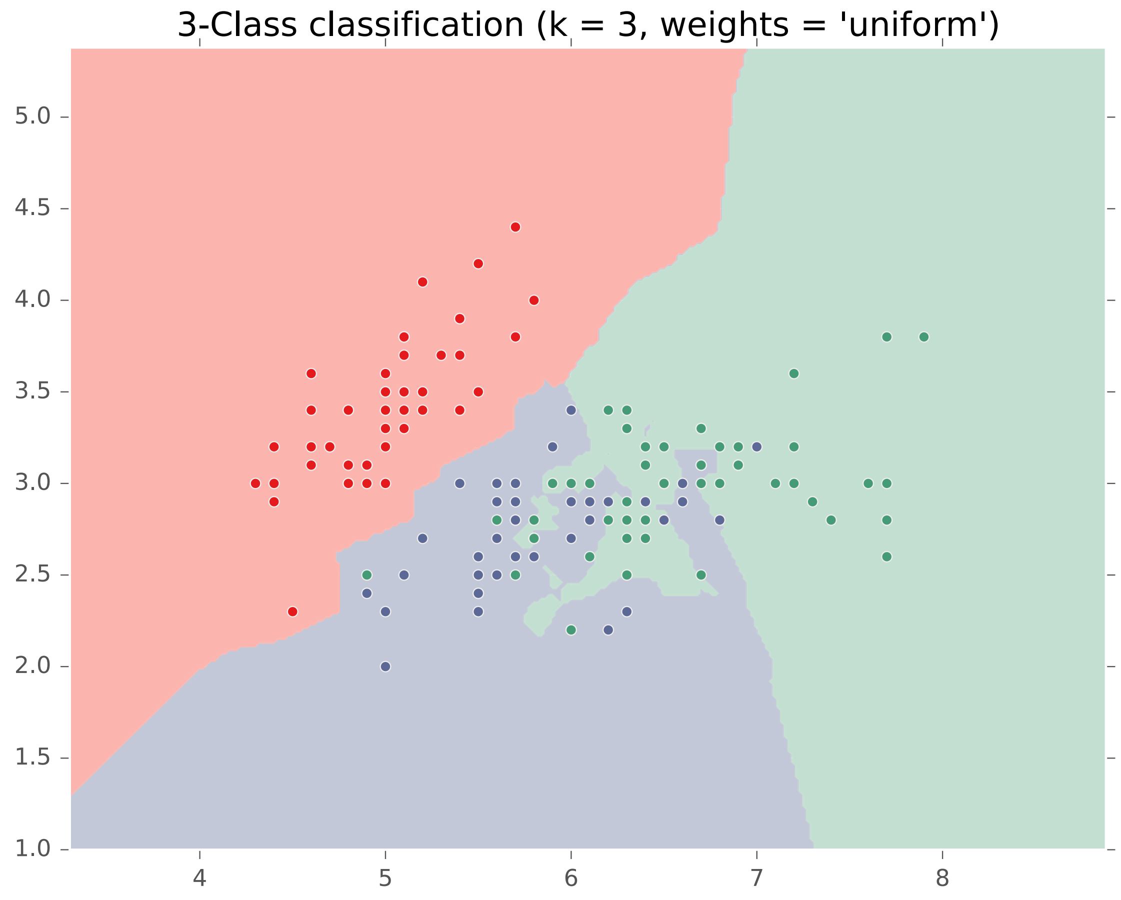


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

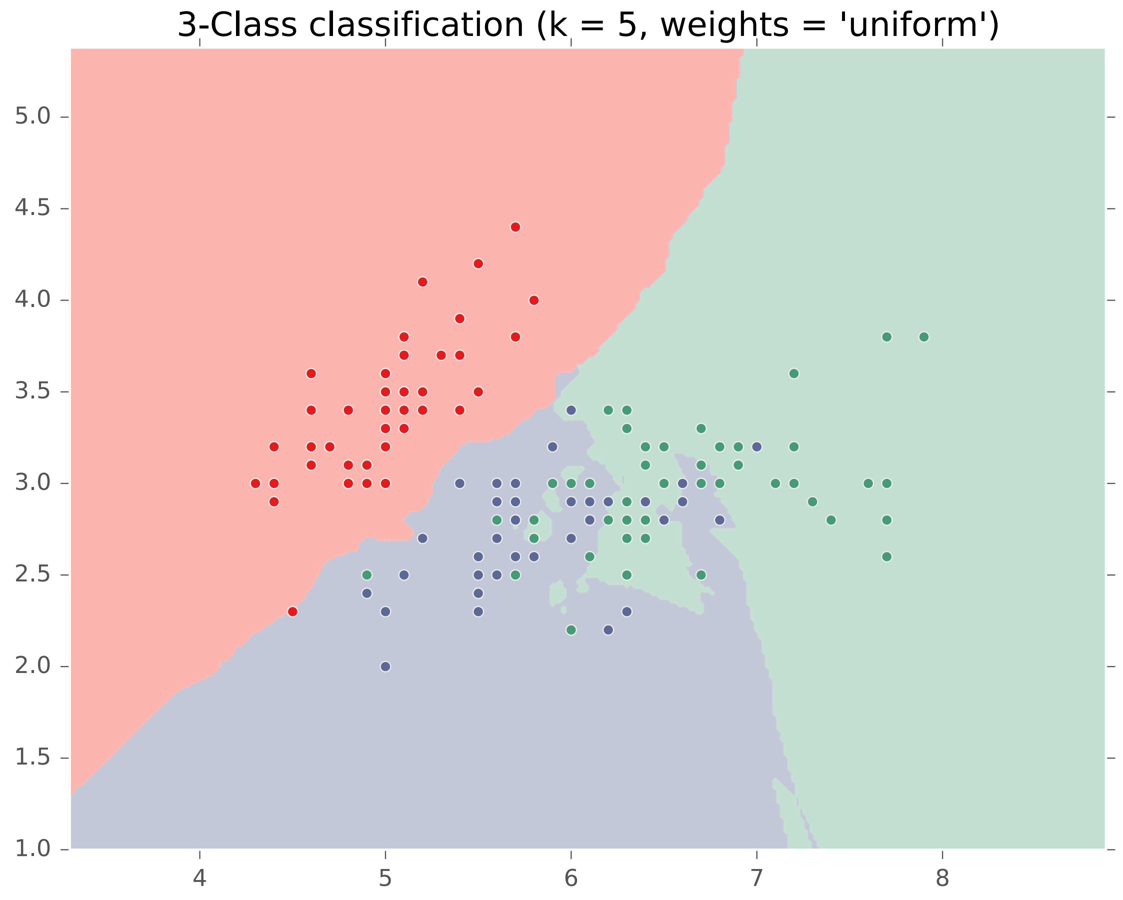


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

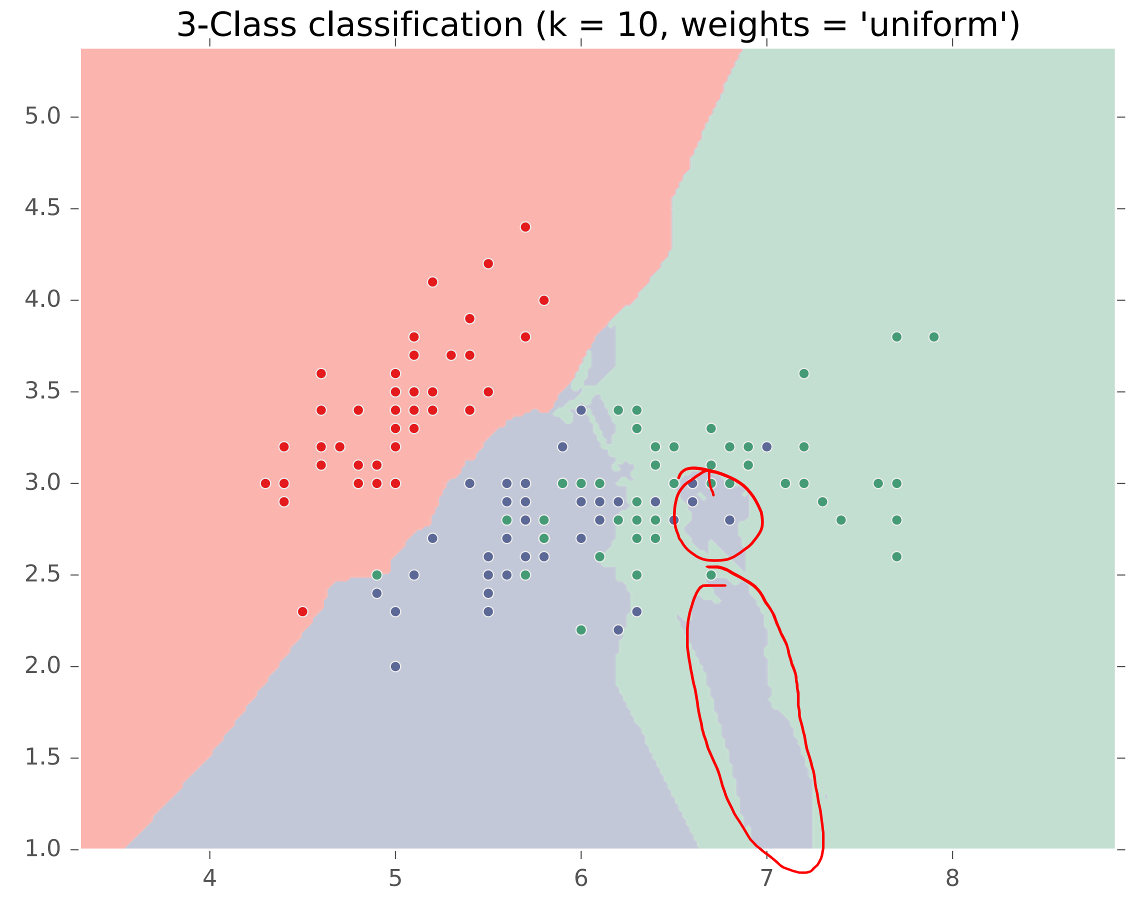


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

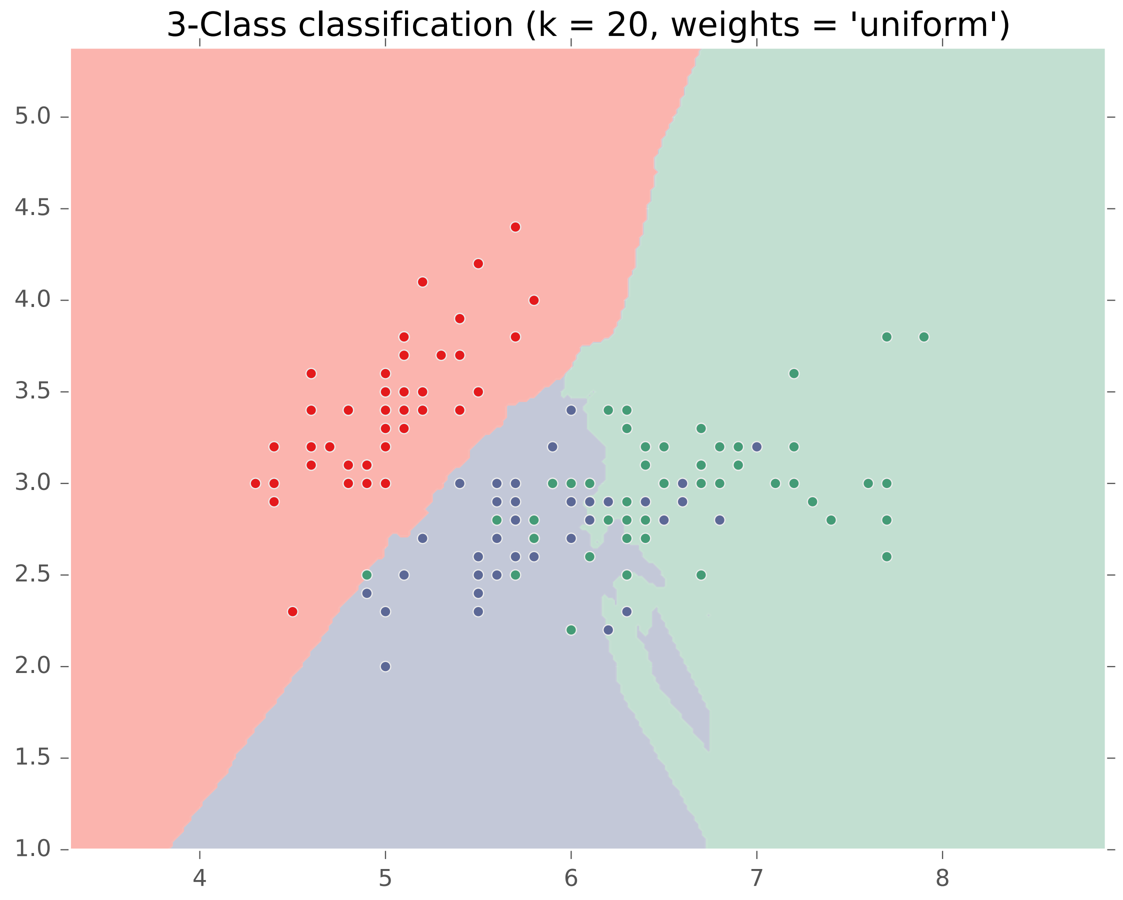


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

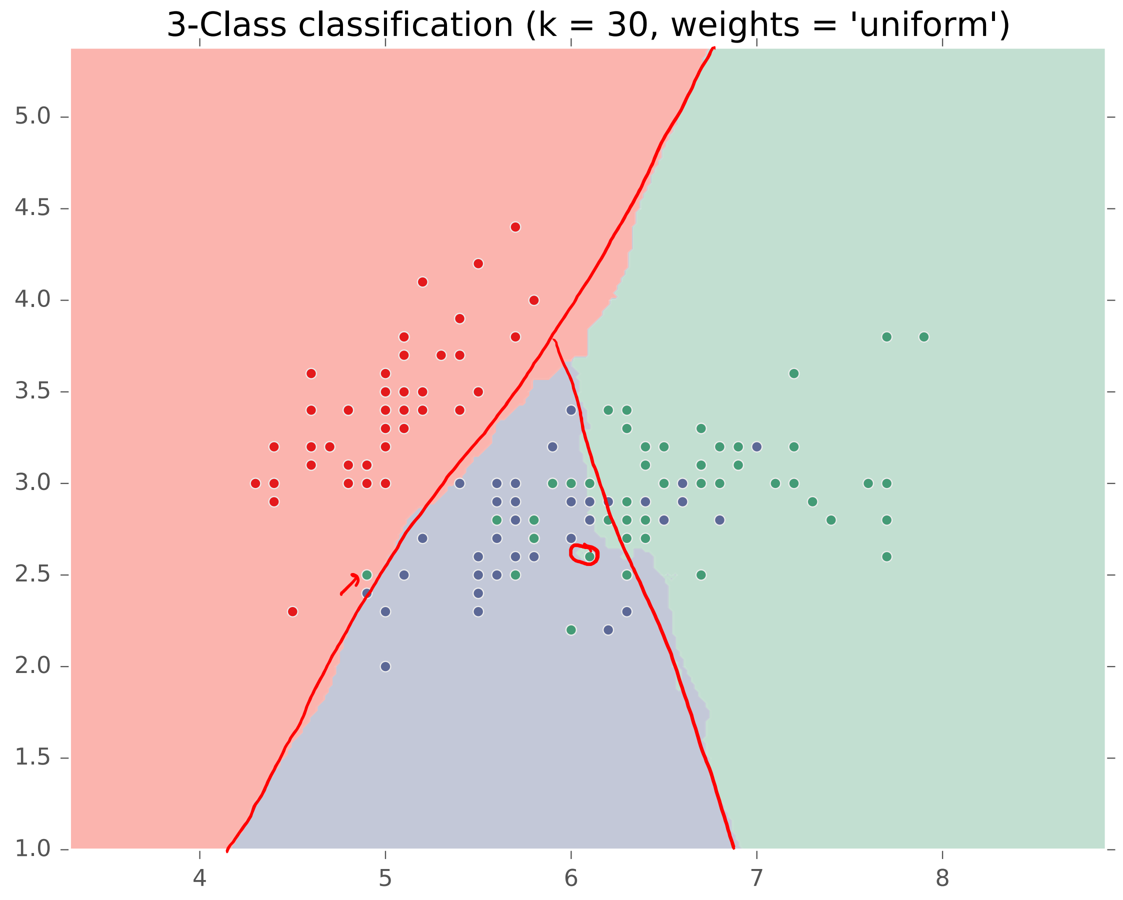


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

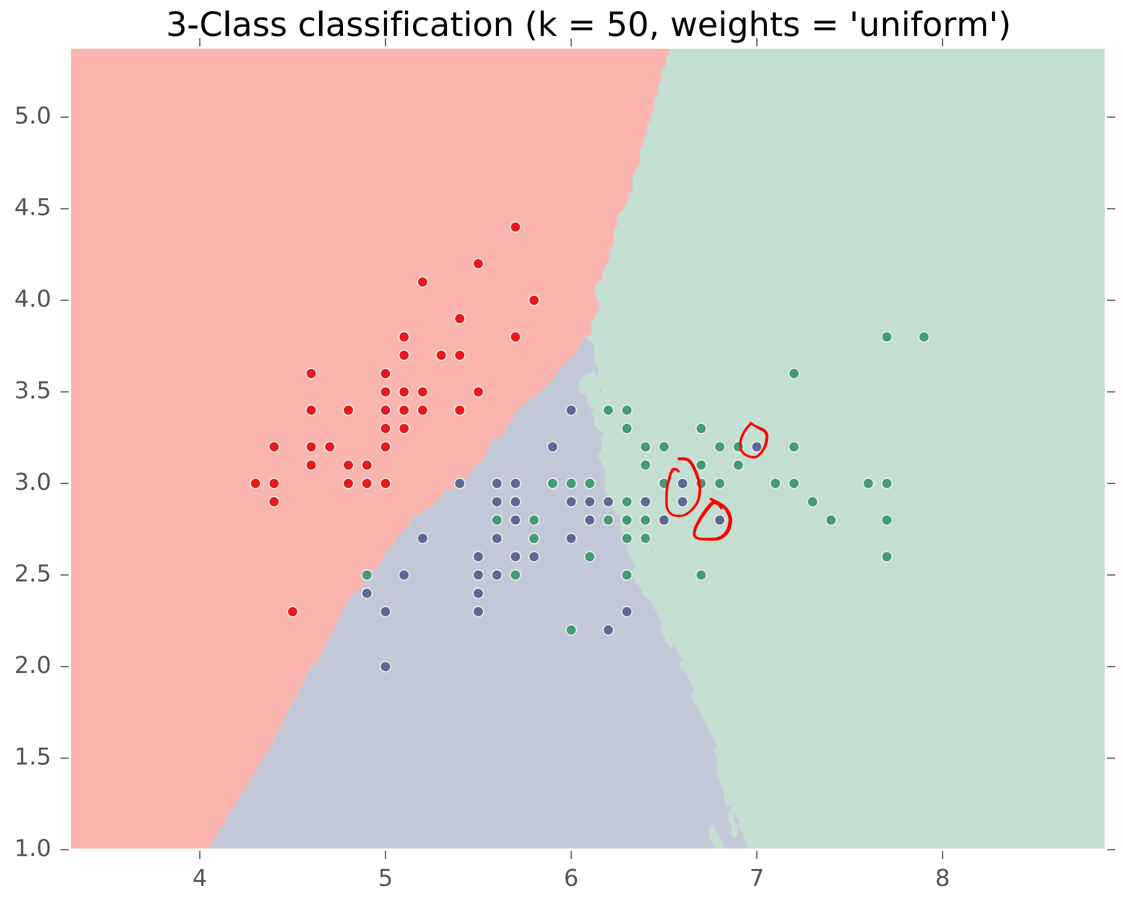


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

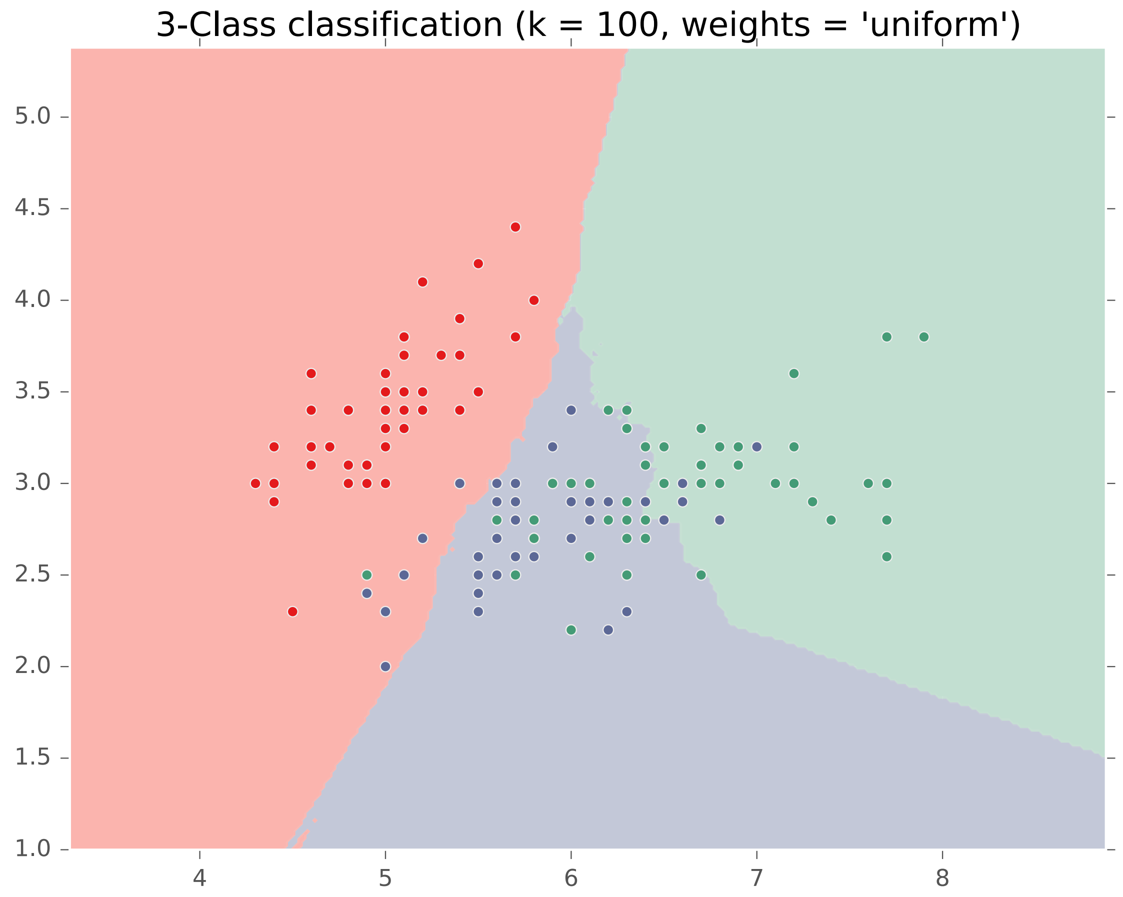


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

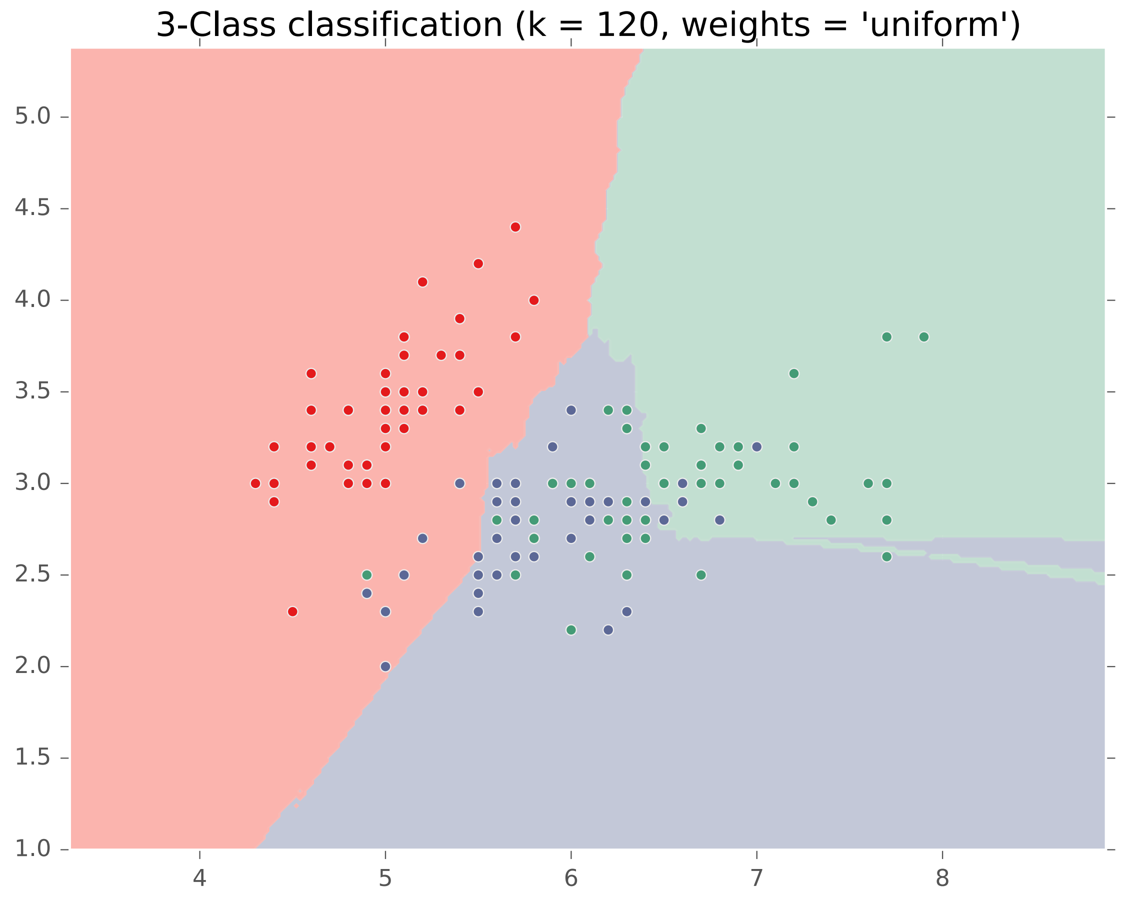


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

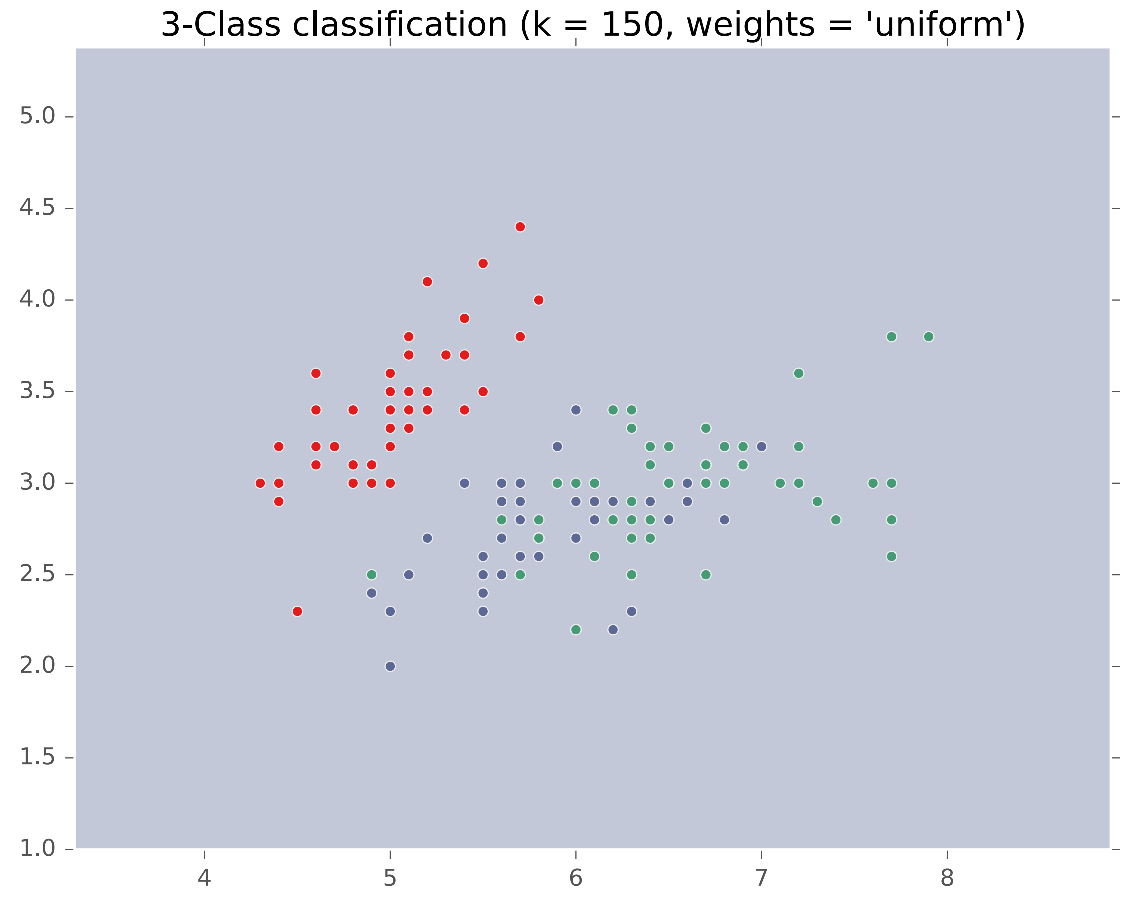


Figure courtesy of Matt Gormley

Setting k

- When $k = 1$:
  - many, complicated decision boundaries
  - may overfit
- When $k = N$:
  - no decision boundaries; always predicts the most common label in the training data
  - may underfit
- k controls the complexity of the hypothesis set $\Longrightarrow$ k affects how well the learned hypothesis will generalize

Setting k

- Theorem:
  - If k is some function of N s.t. $k(N) \to \infty$ and $\frac{k(N)}{N} \to 0$ as $N \to \infty$ …
  - … then (under certain assumptions) the true error of a kNN model $\to$ the Bayes error rate
- Practical heuristics:
  - $k = \lfloor \sqrt{N} \rfloor$
  - $k = 3$
- Can also set k through (cross-)validation (stay tuned)

Aside: kNN and Categorical Features

- kNNs are compatible with categorical features, either by:

1. Converting categorical features into binary ones:

Cholesterol
Normal
Normal
Abnormal

$\longrightarrow$

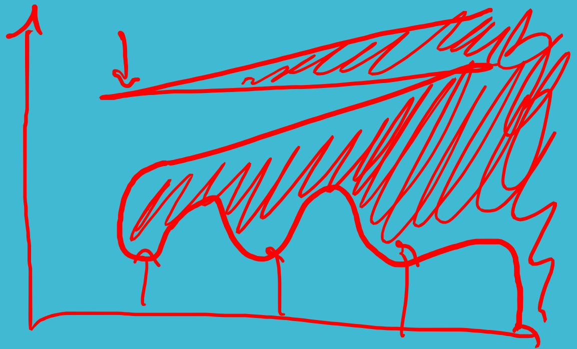
Normal Cholesterol?	Abnormal Cholesterol?
1	0
1	0
0	1

2. Using a distance metric that works over categorical features e.g., the Hamming distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \mathbb{1}(x_d \neq x'_d)$$

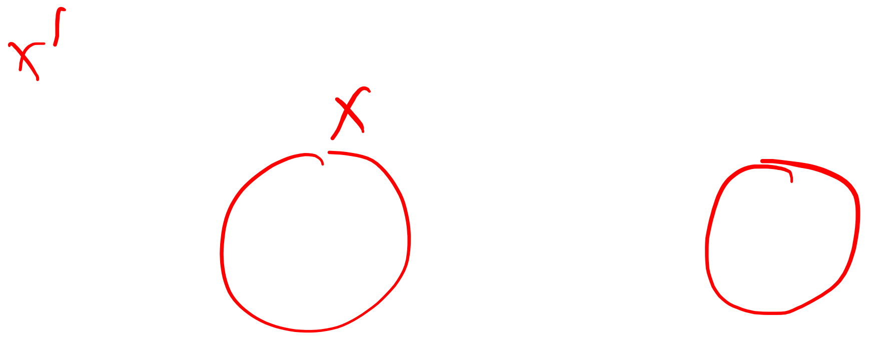
$$d(x, x') = \sqrt{2^2 + 3^2}$$

$$d(x_{new}, x'_{new}) = \sqrt{2^2 + (0.03)^2} \approx \sqrt{4} = 2$$



kNN: Inductive Bias

- Similar points should have similar labels and *all features are equivalently important for determining similarity*



- Feature scale can dramatically influence results!

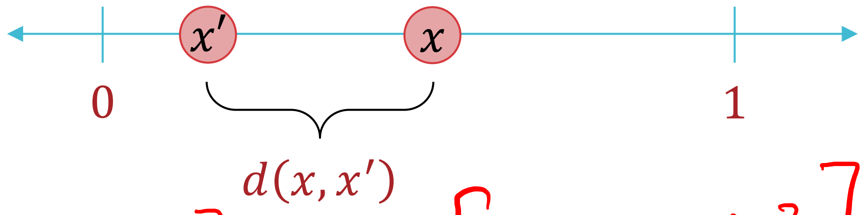
Figure courtesy of Matt Gormley

Curse of Dimensionality

- The fundamental assumption of kNN is that “similar” points or points close to one another should have the same label
- The closer two points are, the more confident we can be that they will have the same label
- As the dimensionality of the input grows, the less likely it is that two random points will be close
- As the dimensionality of the input grows, it takes more points to “cover” the input space

Curse of Dimensionality

- Suppose you independently draw two one-dimensional points between 0 and 1 uniformly at random:



$$E[d(x,x')^2] = E[(x-x')^2]$$

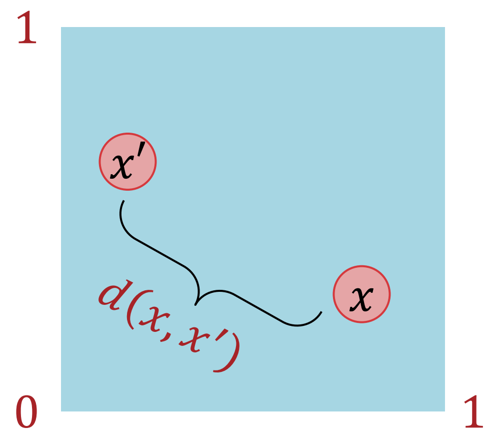
$$= E[x^2 - 2xx' + x'^2]$$

$$= E[x^2] - 2\underbrace{E[xx']} + E[x'^2]$$

$$= E[x^2] - 2E[x]E[x'] + E[x'^2] = \frac{1}{3} - 2\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \frac{1}{3} = 1/6$$

Curse of Dimensionality

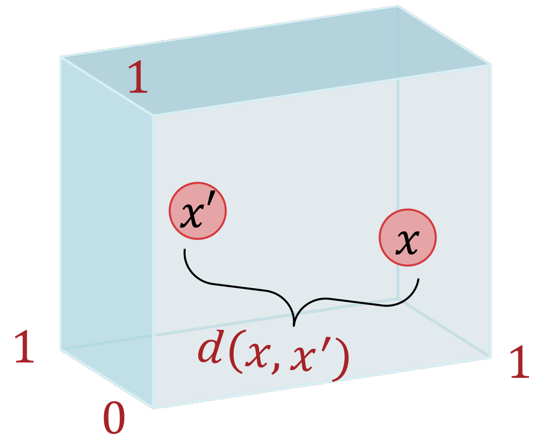
- Suppose you independently draw two two-dimensional points in the unit square uniformly at random:



$$E[d(x,x')^2] = E[(x_1 - x_1')^2 + (x_2 - x_2')^2]$$
$$= E[(x_1 - x_1')^2] + E[(x_2 - x_2')^2]$$
$$= \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$

Curse of Dimensionality

- Suppose you independently draw two three-dimensional points in the unit cube uniformly at random:



$$E[d(x,x')^2] = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

Curse of Dimensionality

- Assume all dimensions of the input are independent and identically distributed.
- Given $N + 1$ data points, $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}$ and $\boldsymbol{x}^*$, let

$$d_+ = \max_{\boldsymbol{x} \in \mathcal{D}} d(\boldsymbol{x}, \boldsymbol{x}^*) \text{ and } d_- = \min_{\boldsymbol{x} \in \mathcal{D}} d(\boldsymbol{x}, \boldsymbol{x}^*)$$

- Then

$$\lim_{D \to \infty} \mathbb{E}\left[\frac{d_+ - d_-}{d_-}\right] \to 0$$

Curing the Curse of Dimensionality

- More data
- Fewer dimensions
- Blessing of non-uniformity: data from the real world is rarely uniformly distributed across the input space

kNN: Pros and Cons

- Pros:
  - Intuitive / explainable
  - No training / retraining
  - Provably near-optimal in terms of true error rate
- Cons:
  - Computationally expensive
    - Always needs to store all data: $O(ND)$
    - Finding the k closest points in D dimensions: $O(ND + N\log(k))$
      - Can be sped up through clever use of data structures (trades off training and test costs)
      - Can be approximated using stochastic methods
  - Affected by feature scale
  - Suffers from the curse of dimensionality

Key Takeaways

- Real-valued features and decision boundaries
- Nearest neighbor model and generalization guarantees
- kNN "training" and prediction
- Effect of k on model complexity
- kNN inductive bias
- Curse of dimensionality