Q & A:

What do we predict in leaf nodes with no training data?

- Well, there isn’t really a majority label, so (if forced to) we could return a random label or a majority vote over the entire training dataset.

- This is related to the question of “what should we predict if some feature in our test dataset takes on a value we didn’t observe in the training dataset?”
  - Going down a branch corresponding to an unseen feature value is like hitting a leaf node with no training data.
Q & A:

When training decision trees, what can we do if the set of values a feature can take on is really large or even infinite?

- If your features are real-valued, there’s a clever trick that only considers $O(L)$ splits where $L$ = the # of values the feature takes on in the training set. Can you guess what it does?
Announcements:

- HW2 released 9/7, due 9/19 at 11:59 PM
- Check out Lecture 3.5, a mini-lecture on how to recursively train decision trees.
- HW1 exit poll released 9/8, you must respond by 9/15 in order to receive full credit.
Q & A:

Wait, I thought you said the exit poll wouldn’t come out until a few days after HW1 was due?

- Ah yes, that’s my bad: I forgot that we use your responses to the exit poll to guide our solution sessions.
- Follow-up: what’s a “solution session”?
  - Oh, great question! Instead of releasing solutions to our HWs, our TAs will go through the solutions with you all (using your feedback from the exit polls) and field any questions you may have.
  - These sessions will be live-streamed and recorded but the recording will only be available for 24 hours.
- All the details for HW1’s solution session have been announced on Piazza.
Q & A:

Speaking of TAs...

- One of your TAs is my [lab partner, teammate, best friend, father’s brother’s nephew’s cousin’s former roommate, etc...]: is it alright if I bug them extra help on [LinkedIn, Instagram, the bus, their way home, etc...]?  

- NO, ABSOLUTELY NOT!

- You may only interact with our TAs about course content during course sanctioned events (e.g., OH or recitations) or on course sanctioned platforms (e.g., Piazza).
Mutual information describes how much information or clarity a particular feature provides about the label

\[ I(Y; x_d) = H(Y) - \sum_{v \in V(x_d)} (f_v)(H(Y_{x_d=v})) \]

where \( x_d \) is a feature

- \( Y \) is the collection of all labels
- \( V(x_d) \) is the set of unique values of \( x_d \)
- \( f_v \) is the fraction of inputs where \( x_d = v \)
- \( Y_{x_d=v} \) is the collection of labels where \( x_d = v \)
- \( H(Y_{x_d=v}) \) is the conditional entropy of \( Y \) given \( x_d = v \)
Mutual information describes how much information or clarity a particular feature provides about the label $Y$.

$$I(Y; x_d) = H(Y) - \sum_{v \in V(x_d)} (f_v)(H(Y|x_d = v))$$

where $x_d$ is a feature

- $Y$ is the collection of all labels
- $V(x_d)$ is the set of unique values of $x_d$
- $f_v$ is the fraction of inputs where $x_d = v$
- $H(Y|x_d = v)$ is the conditional entropy of $Y$ given $x_d = v$ i.e., the entropy of all labels corresponding to examples where $x_d = v$
Mutual information describes how much information or clarity a particular feature provides about the label $Y$

$$I(Y; x_d) = H(Y) - H(Y|x_d)$$

where $x_d$ is a feature

- $Y$ is the collection of all labels
- $V(x_d)$ is the set of unique values of $x_d$
- $H(Y|x_d)$ is the conditional entropy of $Y$ given $x_d$
Mutual Information: Example

\[
I(Y; x_d) = H(Y) - \sum_{v \in V(x_d)} (f_v) \left( H(Y_{x_d=v}) \right)
\]

\[
= 1 - \frac{1}{2} H(Y_{x_d=0}) - \frac{1}{2} H(Y_{x_d=1})
\]

\[
= 1 - \frac{1}{2} (0) - \frac{1}{2} (0) = 1
\]
Mutual Information: Example

\[ I(Y; x_d) = H(Y) - \sum_{v \in V(x_d)} (f_v) \left( H(Y_{x_d=v}) \right) \]

\[ = 1 - \frac{1}{2} H(Y_{x_d=0}) - \frac{1}{2} H(Y_{x_d=1}) \]

\[ = 1 - \frac{1}{2} (1) - \frac{1}{2} (1) = 0 \]
Poll Question 3:

Which feature would you split on using mutual information as the splitting criterion?

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$
\mathbb{I}(Y; x) = H(Y) - 0 \left[ H(Y_{x=0}) - 1 \left( H(Y_{x=1}) \right) \right] = 0
$$

$$
\begin{align*}
\mathbb{I}(Y; x_2) &= H(Y) - \frac{1}{2} \left( H(Y_{x_2=0}) - \frac{1}{2} \left( H(Y_{x_2=1}) \right) \right) \\
 &= -\frac{2}{8} \log_2 \frac{2}{8} - \frac{6}{8} \log_2 \frac{6}{8}
\end{align*}
$$
Bluntine & Niblett (1992)
Splitting Criteria Experiments

Table 3. Error for different splitting rules (pruned trees).

<table>
<thead>
<tr>
<th>Data Set</th>
<th>GINI</th>
<th>Info. Gain</th>
<th>Marsh.</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypo</td>
<td>1.01 ± 0.29</td>
<td>0.95 ± 0.22</td>
<td>1.27 ± 0.47</td>
<td>7.44 ± 0.53</td>
</tr>
<tr>
<td>breast</td>
<td>28.66 ± 3.87</td>
<td>28.49 ± 4.28</td>
<td>27.15 ± 4.22</td>
<td>29.65 ± 4.97</td>
</tr>
<tr>
<td>tumor</td>
<td>60.88 ± 5.44</td>
<td>62.70 ± 3.89</td>
<td>61.62 ± 3.98</td>
<td>67.94 ± 5.68</td>
</tr>
<tr>
<td>lymph</td>
<td>24.44 ± 6.92</td>
<td>24.00 ± 6.87</td>
<td>24.33 ± 5.51</td>
<td>32.33 ± 11.25</td>
</tr>
<tr>
<td>LED</td>
<td>33.77 ± 3.06</td>
<td>32.89 ± 2.59</td>
<td>33.15 ± 4.02</td>
<td>38.18 ± 4.57</td>
</tr>
<tr>
<td>mush</td>
<td>1.44 ± 0.47</td>
<td>1.44 ± 0.47</td>
<td>7.31 ± 2.25</td>
<td>8.77 ± 4.65</td>
</tr>
<tr>
<td>votes</td>
<td>4.47 ± 0.95</td>
<td>4.57 ± 0.87</td>
<td>11.77 ± 3.95</td>
<td>12.40 ± 4.56</td>
</tr>
<tr>
<td>votes1</td>
<td>12.79 ± 1.48</td>
<td>13.04 ± 1.65</td>
<td>15.13 ± 2.89</td>
<td>15.62 ± 2.73</td>
</tr>
<tr>
<td>iris</td>
<td>5.00 ± 3.08</td>
<td>4.90 ± 3.08</td>
<td>5.50 ± 2.59</td>
<td>14.20 ± 6.77</td>
</tr>
<tr>
<td>glass</td>
<td>39.56 ± 6.20</td>
<td>50.57 ± 6.73</td>
<td>40.53 ± 6.41</td>
<td>53.20 ± 5.01</td>
</tr>
<tr>
<td>xd6</td>
<td>22.14 ± 3.23</td>
<td>22.17 ± 3.36</td>
<td>22.06 ± 3.37</td>
<td>31.86 ± 3.62</td>
</tr>
<tr>
<td>pole</td>
<td>15.43 ± 1.51</td>
<td>15.47 ± 0.88</td>
<td>15.01 ± 1.15</td>
<td>26.38 ± 6.92</td>
</tr>
</tbody>
</table>

Key takeaway: Gini impurity and Information gain (aka mutual information) are nearly identical.
### Table 4. Difference and significance of error for GINI splitting rule versus others.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Info. Gain</th>
<th>Marsh.</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypo</td>
<td>-0.06 (0.82)</td>
<td>0.26 (0.99)</td>
<td>6.43 (1.00)</td>
</tr>
<tr>
<td>breast</td>
<td>-0.17 (0.23)</td>
<td>-1.51 (0.94)</td>
<td>0.99 (0.72)</td>
</tr>
<tr>
<td>tumor</td>
<td>1.81 (0.84)</td>
<td>0.74 (0.39)</td>
<td>7.06 (0.99)</td>
</tr>
<tr>
<td>lymph</td>
<td>-0.44 (0.83)</td>
<td>-0.11 (0.05)</td>
<td>7.89 (0.99)</td>
</tr>
<tr>
<td>LED</td>
<td>0.12 (0.17)</td>
<td>0.38 (0.41)</td>
<td>5.41 (0.99)</td>
</tr>
<tr>
<td>mush</td>
<td>0.00 (0.00)</td>
<td>5.86 (1.00)</td>
<td>7.32 (0.99)</td>
</tr>
<tr>
<td>votes</td>
<td>0.11 (0.55)</td>
<td>7.30 (0.99)</td>
<td>7.94 (0.99)</td>
</tr>
<tr>
<td>votesl</td>
<td>0.26 (0.47)</td>
<td>2.34 (0.98)</td>
<td>2.83 (0.99)</td>
</tr>
<tr>
<td>iris</td>
<td>-0.10 (0.67)</td>
<td>0.50 (0.90)</td>
<td>9.20 (0.99)</td>
</tr>
<tr>
<td>glass</td>
<td>1.01 (0.50)</td>
<td>0.96 (0.53)</td>
<td>13.64 (0.99)</td>
</tr>
<tr>
<td>xd6</td>
<td>0.04 (0.11)</td>
<td>-0.07 (0.20)</td>
<td>9.72 (0.99)</td>
</tr>
<tr>
<td>pole</td>
<td>0.03 (0.11)</td>
<td>-0.43 (0.83)</td>
<td>10.95 (0.99)</td>
</tr>
</tbody>
</table>

The average difference in error between the splitting criterion and the difference according to a two-sided paired t-test.
def train(\(\mathcal{D}\)):
    store root = tree_recurse(\(\mathcal{D}\))

def tree_recurse(\(\mathcal{D}'\)):
    q = new node()
    base case – if (SOME CONDITION):
    recursion – else:
        q.type = internal
        find best attribute to split on, \(x_d\)
        q.split = \(x_d\)
        for \(v\) in \(V(x_d)\), all possible values of \(x_d\):
            \(\mathcal{D}_v = \{(x^{(n)}, y^{(n)}) \in \mathcal{D} | x_d^{(n)} = v\}\)
            q.children(\(v\)) = tree_recurse(\(\mathcal{D}_v\))
    return q
def train(D):
    store root = tree_recurse(D)

def tree_recurse(D):
    q = new node()
    base case – if (D' is empty OR
    all labels in D' are the same OR
    all features in D' are identical OR
    some other stopping criterion):
        q.type = leaf
        q.label = majority_vote(labels in D')

    recursion – else:
    return q
Poll Question 1: Which decision tree would you learn if you used training error rate as the splitting criterion? Break ties alphabetically.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

9/12/22
Poll Question 2: Which decision tree is the smallest decision tree that achieves the lowest possible training error?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>−</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>−</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>−</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>−</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>
Decision Trees:
Pros & Cons

- **Pros**
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features

- **Cons**
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Liable to overfit!
Decision Trees: Inductive Bias

- The **inductive bias** of a machine learning algorithm is the principle by which it generalizes to unseen examples.

- What is the inductive bias of the ID3 algorithm?

  Occam's Razor: the simplest explanation for some training dataset is best.

  Return the decision tree with 0 training error by splitting on high mutual information features first.
Overfitting occurs when the classifier (or model)...
- is too complex
- fits noise or “outliers” in the training dataset as opposed to the actual pattern of interest
- doesn’t have enough inductive bias pushing it to generalize

Underfitting occurs when the classifier (or model)...
- is too simple
- can’t capture the actual pattern of interest in the training dataset
- has too much inductive bias

e.g. majority vote
Recall: Different Kinds of Error

- Training error rate = \( \text{err}(h, D_{\text{train}}) \)
- Test error rate = \( \text{err}(h, D_{\text{test}}) \)
- True error rate = \( \text{err}(h) \)
  
  = the error rate of \( h \) on all possible examples

  - In machine learning, this is the quantity that we care about but, in most cases, it is unknowable.

- Overfitting occurs when \( \text{err}(h) > \text{err}(h, D_{\text{train}}) \)

  - \( \text{err}(h) - \text{err}(h, D_{\text{train}}) \) can be thought of as a measure of overfitting
Overfitting in Decision Trees
Combatting Overfitting in Decision Trees

- Heuristics:
  - Do not split leaves past a fixed depth, $\delta$
  - Do not split leaves with fewer than $c$ data points
  - Do not split leaves where the maximal information gain is less than $\tau$
  - Take a majority vote in impure leaves
Combatting Overfitting in Decision Trees

- **Pruning:**
  - First, learn a decision tree
  - Then, evaluate each split using a “validation” dataset by comparing the validation error rate with and without that split
  - Greedily remove the split that most decreases the validation error rate if such a split exists
  - Stop if no split is removed
Pruning Decision Trees

Figure courtesy of Tom Mitchell
You should be able to...

1. Implement decision tree training and prediction
2. Use effective splitting criteria for decision trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in decision tree learning
You should be able to...

1. Implement decision tree training and prediction
2. Use effective splitting criteria for decision trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in decision tree learning

Poll Question 3:
What questions do you have?
Real-valued Features
Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

<table>
<thead>
<tr>
<th>Species</th>
<th>Sepal Length</th>
<th>Sepal Width</th>
<th>Petal Length</th>
<th>Petal Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>4.9</td>
<td>3.6</td>
<td>1.4</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>5.3</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>2.4</td>
<td>3.3</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>5.7</td>
<td>2.8</td>
<td>4.1</td>
<td>1.3</td>
</tr>
<tr>
<td>1</td>
<td>6.3</td>
<td>3.3</td>
<td>4.7</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>6.7</td>
<td>3.0</td>
<td>5.0</td>
<td>1.7</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Species</th>
<th>Sepal Length</th>
<th>Sepal Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.3</td>
<td>3.0</td>
</tr>
<tr>
<td>0</td>
<td>4.9</td>
<td>3.6</td>
</tr>
<tr>
<td>0</td>
<td>5.3</td>
<td>3.7</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>2.4</td>
</tr>
<tr>
<td>1</td>
<td>5.7</td>
<td>2.8</td>
</tr>
<tr>
<td>1</td>
<td>6.3</td>
<td>3.3</td>
</tr>
<tr>
<td>1</td>
<td>6.7</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Fisher Iris Dataset

Figure courtesy of Matt Gormley
The Duck Test

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 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:

\[
d(x, x') = \|x - x'\|_1 = \sum_{d=1}^{D} |x_d - x'_d|
\]
Nearest Neighbor: Pseudocode

```python
def train(D_train):
    store D_train

def predict(x_prime):
    find the closest point to \( x' \) in \( D_{train} \)

    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 $n \to \infty$, 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.”

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 $\leq 2 \times$ 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 (\(k\)NN)

- 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)
  - Weight votes by distance
  - Remove furthest neighbor
  - Add next closest neighbor
  - Use a different distance metric
\( k \)-Nearest Neighbors (kNN): Pseudocode

def train(\( \mathcal{D} \)):
    store \( \mathcal{D} \)

def predict(\( x' \)):
    return majority_vote(labels of the \( k \) nearest neighbors to \( x' \) in \( \mathcal{D} \))
$k$NN on Fisher Iris Data

Figure courtesy of Matt Gormley
$k$NN on Fisher Iris Data
$k$NN on Fisher Iris Data
$k$NN on Fisher Iris Data

Figure courtesy of Matt Gormley
$k$NN on Fisher Iris Data
$k$NN on Fisher Iris Data

3-Class classification ($k = 20$, weights = 'uniform')
$k$NN on Fisher Iris Data
$k$NN on Fisher Iris Data

3-Class classification ($k = 50$, weights = 'uniform')

Figure courtesy of Matt Gormley
$k$NN on Fisher Iris Data
$k$NN on Fisher Iris Data

3-Class classification ($k = 120$, weights = 'uniform')
$k$NN on Fisher Iris Data
• Similar points should have similar labels and all features are equivalently important for determining similarity.

• Feature scale can dramatically influence results!
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
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 $\implies k$ affects how well the learned hypothesis will generalize
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 $k$NN model $\to$ the Bayes error rate

Practical heuristics:

- $k = \lfloor \sqrt{N} \rfloor$
- $k = 3$
- Can also set $k$ through (cross-)validation
Curse of Dimensionality

- The fundamental assumption of $k$NN 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.
• Suppose you independently draw two one-dimensional points between 0 and 1 uniformly at random:

\[
\mathbb{E}[d(x, x')] = \mathbb{E}[(x - x')^2] = \mathbb{E}[x^2] - 2\mathbb{E}[x]\mathbb{E}[x'] + \mathbb{E}[x'^2] = 2\mathbb{E}[x^2] - 2\mathbb{E}[x]^2 = 2 \left(\frac{1}{3}\right) - 2 \left(\frac{1}{2}\right)^2 = \frac{1}{6}
\]
• Suppose you independently draw two two-dimensional points in the unit square uniformly at random:

\[ \mathbb{E}[d(x, x')] = \mathbb{E}[(x_1 - x'_1)^2 + (x_2 - x'_2)^2] \]

\[ = 2\mathbb{E}[(x_1 - x'_1)^2] \]

\[ = 2 \left( \frac{1}{6} \right) = \frac{1}{3} \]
• Suppose you independently draw two three-dimensional points in the unit cube uniformly at random:

\[
\mathbb{E}[d(x, x')] = \mathbb{E}[(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + (x_3 - x'_3)^2]
\]
\[
= 3 \mathbb{E}[(x_1 - x'_1)^2]
\]
\[
= 3 \left( \frac{1}{6} \right) = \frac{1}{2}
\]
Curse of Dimensionality

- Assume all dimensions of the input are i.i.d.
  \[ p(x) = \prod_{d=1}^{D} p(x_d) \]

- Given \( N + 1 \) data points, \( \mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\} \) and \( x^* \), all drawn iid from the distribution above, let
  \[ d_+ = \max_{x \in \mathcal{D}} d(x, x^*) \text{ and } d_- = \min_{x \in \mathcal{D}} d(x, x^*) \]

- 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