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

Hoda Heidari, Henry Chai & Matt Gormley

1/29/24

Front Matter

- Announcements:
  - HW2 released 1/24, due 2/5 at 11:59 PM

Recall: Decision Tree – Pseudocode

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

Recall: Decision Tree – Pseudocode

```
def train($\mathcal{D}_{train}$):
    store root = tree_recurse($\mathcal{D}_{train}$)
def tree_recurse($\mathcal{D}'$):
    q = new node()
    base case – if ($\mathcal{D}'$ is empty OR
        all labels in $\mathcal{D}'$ are the same OR
        all features in $\mathcal{D}'$ are identical OR
        some other stopping criterion):
        q.label = majority_vote($\mathcal{D}'$)


    recursion – else:
    return q
```

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 i.e., decision tree learning with mutual information maximization as the splitting criterion?
  - Try to find the __________ tree that achieves ________________________________ with ____________________________ features at the top

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

Decision Trees & Real-Valued Features

x Resting Systolic Pressure	y Heart Disease?
135	No
142	Yes
118	No
147	Yes
121	Yes
133	No
140	No
125	Yes
150	Yes
126	No

→

x Resting Systolic Pressure	y Heart Disease?
118	No
121	Yes
125	Yes
126	No
133	No
135	No
140	No
142	Yes
147	Yes
150	Yes

← $x < 119.5?$

Decision Trees & Real-Valued Features

x Resting Systolic Pressure	y Heart Disease?
135	No
142	Yes
118	No
147	Yes
121	Yes
133	No
140	No
125	Yes
150	Yes
126	No

x Resting Systolic Pressure	y Heart Disease?
118	No
121	Yes
125	Yes
126	No
133	No
135	No
140	No
142	Yes
147	Yes
150	Yes

$x < 123?$

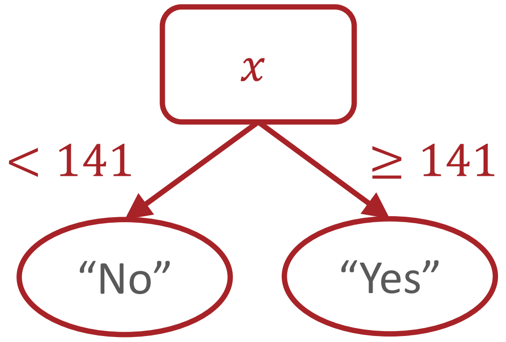
Decision Trees & Real-Valued Features

x Resting Systolic Pressure	y Heart Disease?
135	No
142	Yes
118	No
147	Yes
121	Yes
133	No
140	No
125	Yes
150	Yes
126	No

→

x Resting Systolic Pressure	y Heart Disease?
118	No
121	Yes
125	Yes
126	No
133	No
135	No
140	No
142	Yes
147	Yes
150	Yes

← $x < 141?$



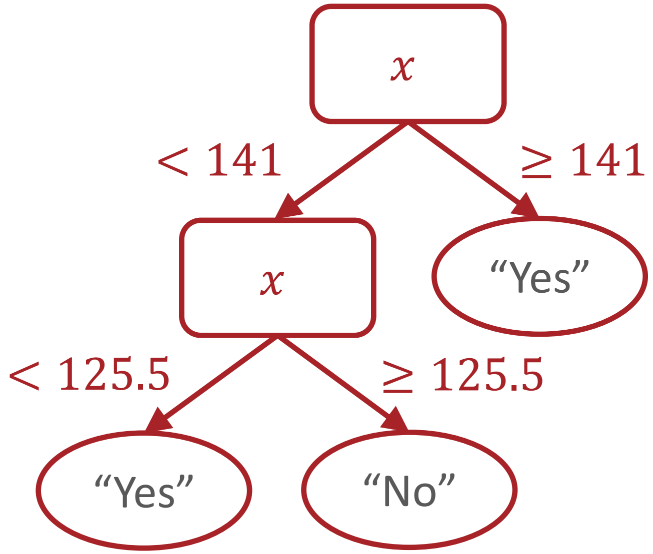
Decision Trees & Real-Valued Features

x Resting Systolic Pressure	y Heart Disease?
135	No
142	Yes
118	No
147	Yes
121	Yes
133	No
140	No
125	Yes
150	Yes
126	No

x Resting Systolic Pressure	y Heart Disease?
118	No
121	Yes
125	Yes
126	No
133	No
135	No
140	No
142	Yes
147	Yes
150	Yes

$x < 125.5?$

$x < 141?$


x

< 141

≥ 141

x

"Yes"

< 125.5

≥ 125.5

"Yes"

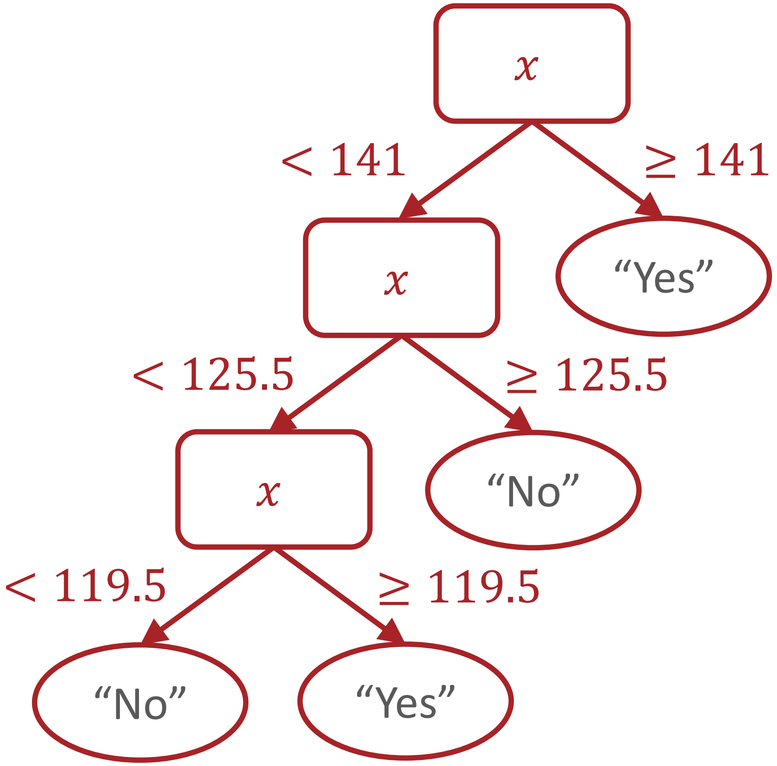
"No"

Decision Trees & Real-Valued Features

x Resting Systolic Pressure	y Heart Disease?
135	No
142	Yes
118	No
147	Yes
121	Yes
133	No
140	No
125	Yes
150	Yes
126	No

→

x Resting Systolic Pressure	y Heart Disease?
118	No
121	Yes
125	Yes
126	No
133	No
135	No
140	No
142	Yes
147	Yes
150	Yes



- Discretize real-valued features using thresholds (effectively creating *new* categorial features)
- Can split on real-valued features multiple times in the same tree

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!

Overfitting

- 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 (e.g., the memorizer)
- 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 = $err(h, \mathcal{D}_{train})$
- Test error rate = $err(h, \mathcal{D}_{test})$
- True error rate = $err(h)$

  = 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 $err(h) > err(h, \mathcal{D}_{train})$
  - $err(h) - err(h, \mathcal{D}_{train})$ can be thought of as a measure of overfitting

Overfitting in Decision Trees

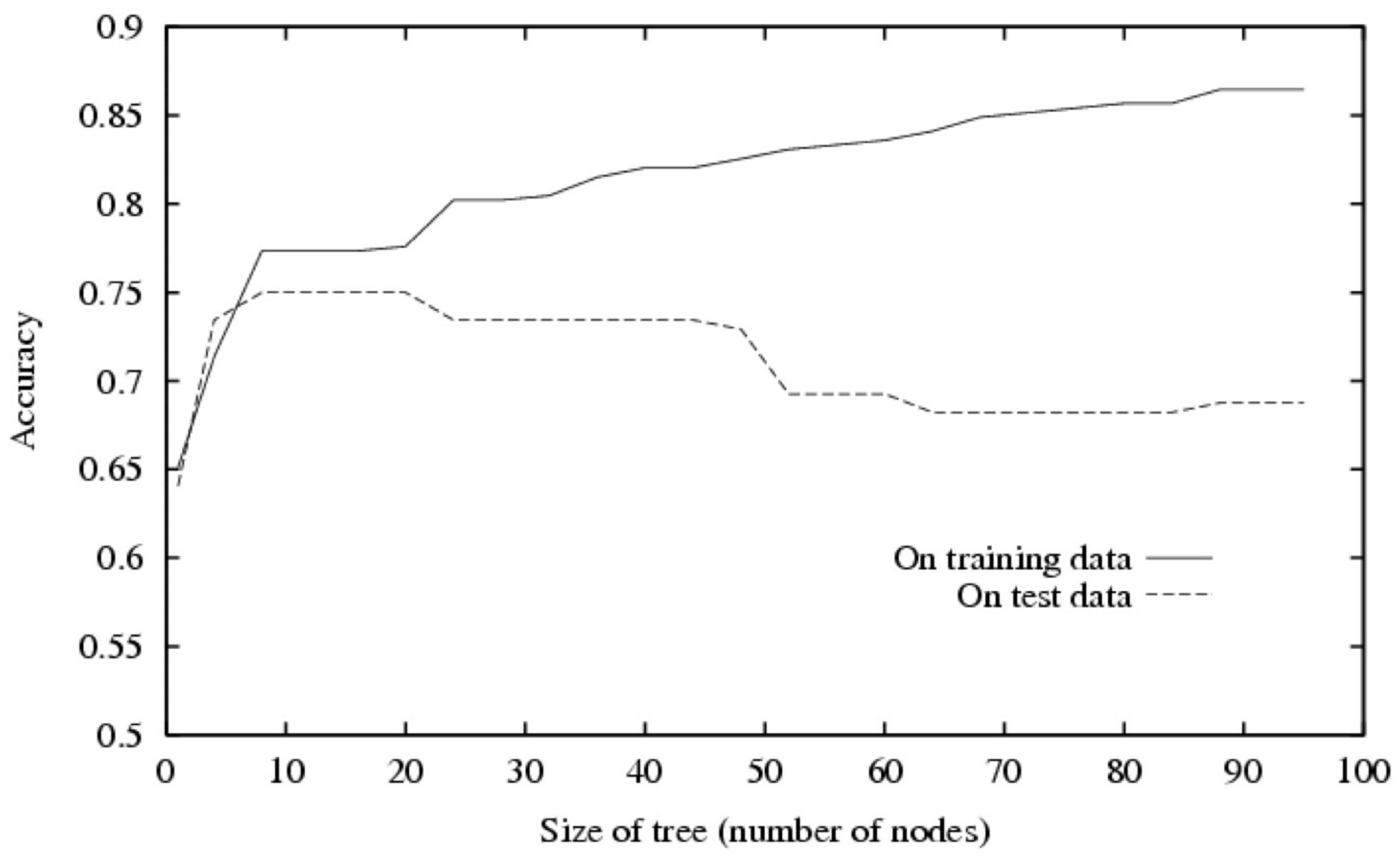


Figure courtesy of Tom Mitchell

Combatting Overfitting in Decision Trees

- Heuristics:
  - Do not split leaves past a fixed depth, δ
  - Do not split leaves with fewer than c data points
  - Do not split leaves where the maximal information gain is less than τ
- 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
  - Stop if no split is removed

Pruning Decision Trees

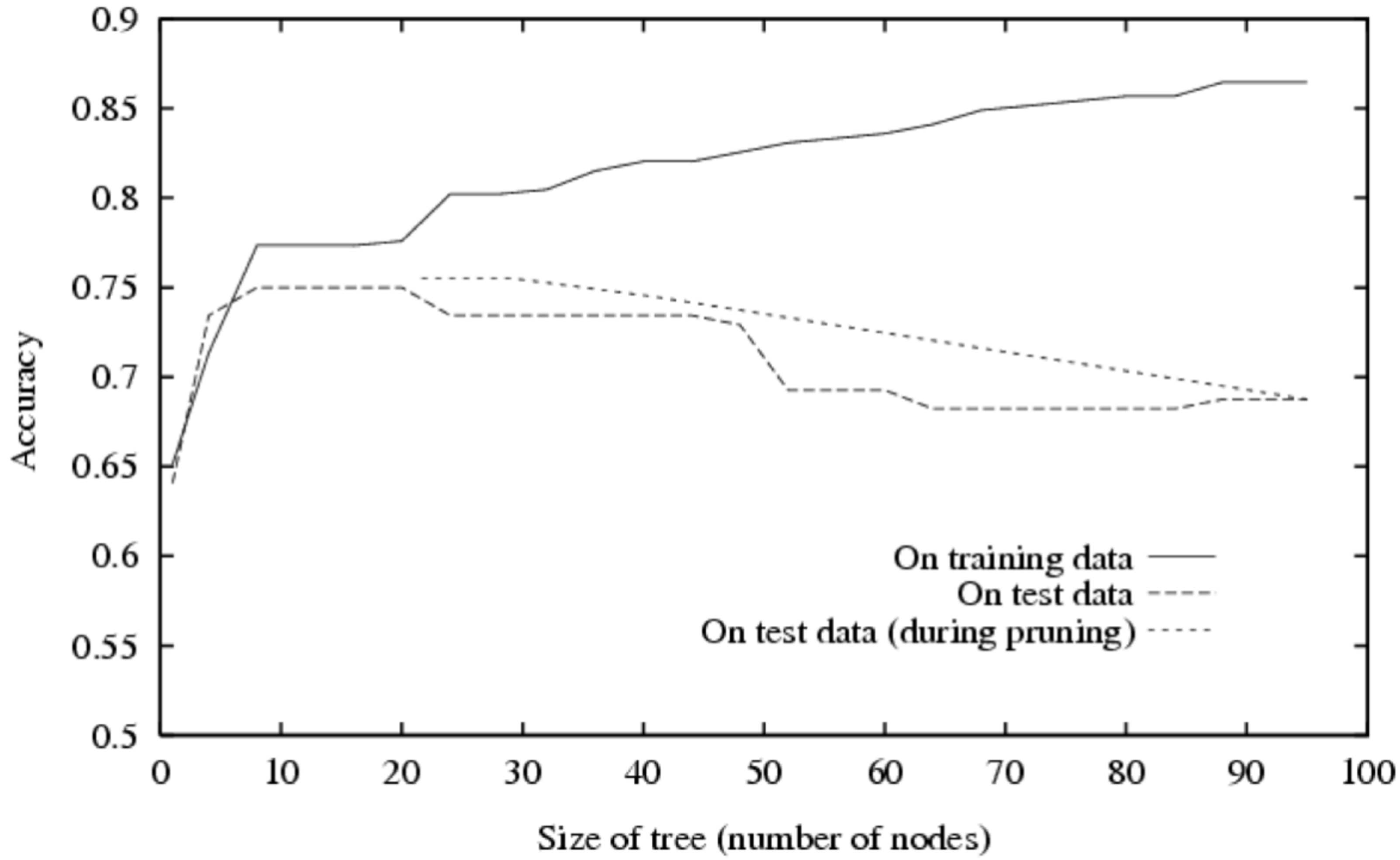


Figure courtesy of Tom Mitchell

Decision Tree Learning Objectives

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

REAL VALUED ATTRIBUTES


sepal
petal

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

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

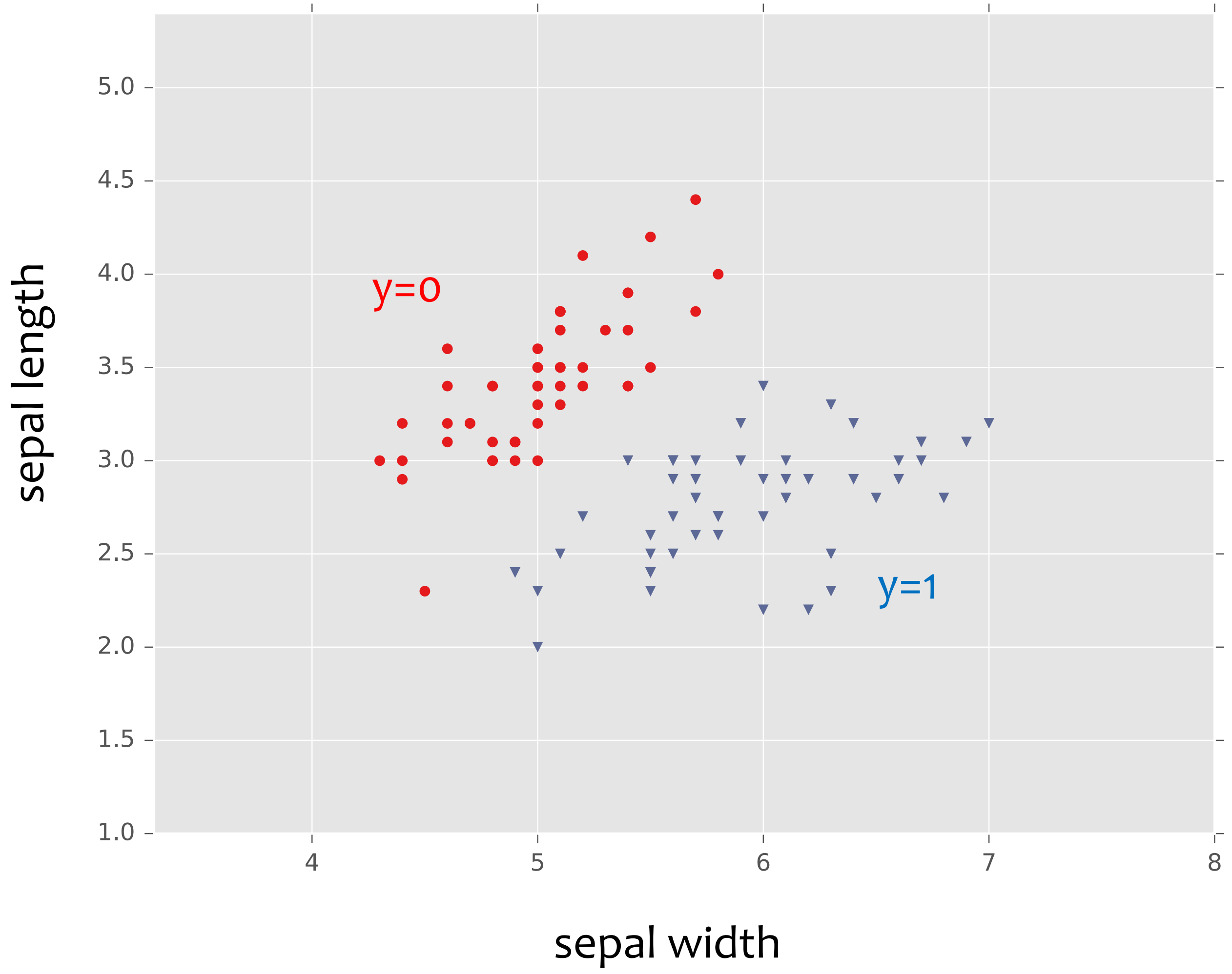
Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

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

Deleted two of the four features, so that input space is 2D

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset



K-NEAREST NEIGHBORS

Nearest Neighbor: Algorithm

```
def train($\mathcal{D}$):
    Store $\mathcal{D}$

def h($\boldsymbol{x}'$):
    Let $\boldsymbol{x}^{(i)}$ = the point in $\mathcal{D}$ that is nearest to $\boldsymbol{x}'$
    return $y^{(i)}$
```

Classification & Real-Valued Features

Classification

Binary Classification

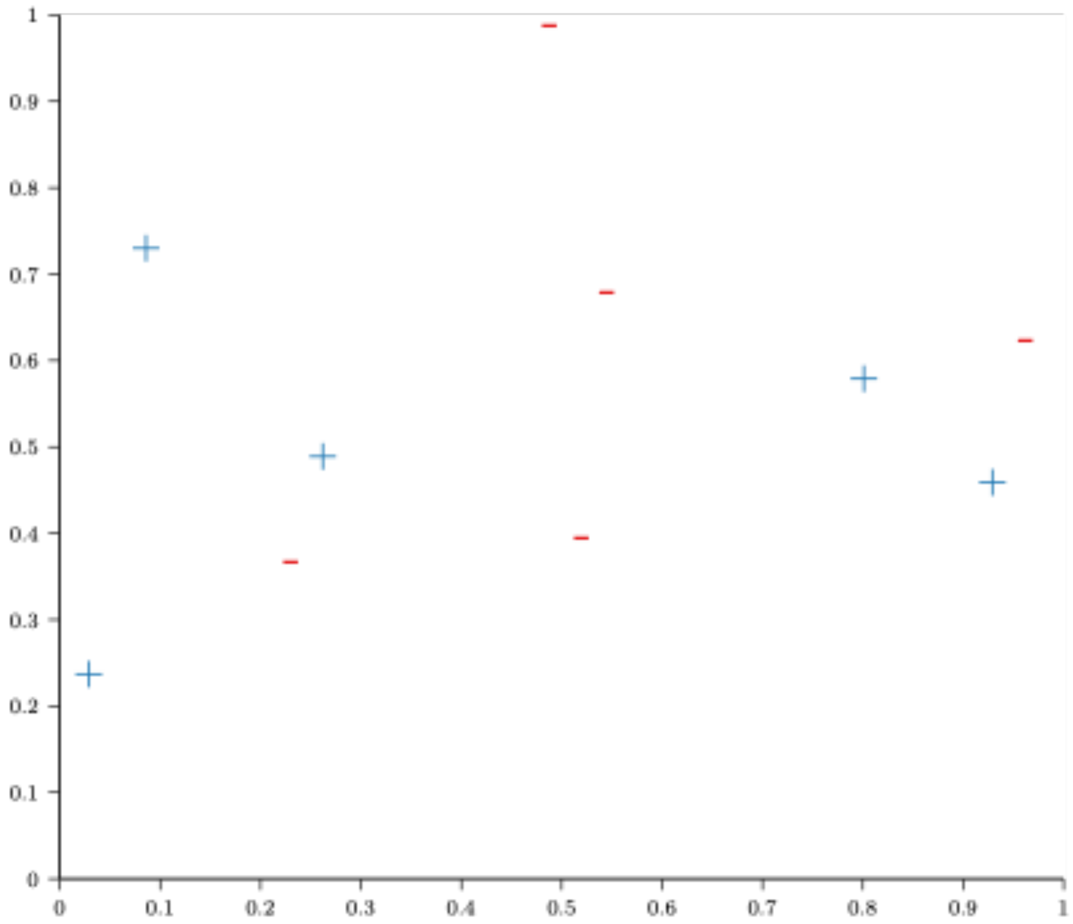
Classification & Real-Valued Features

Decision Rules / Decision Boundaries

Nearest Neighbor: Algorithm

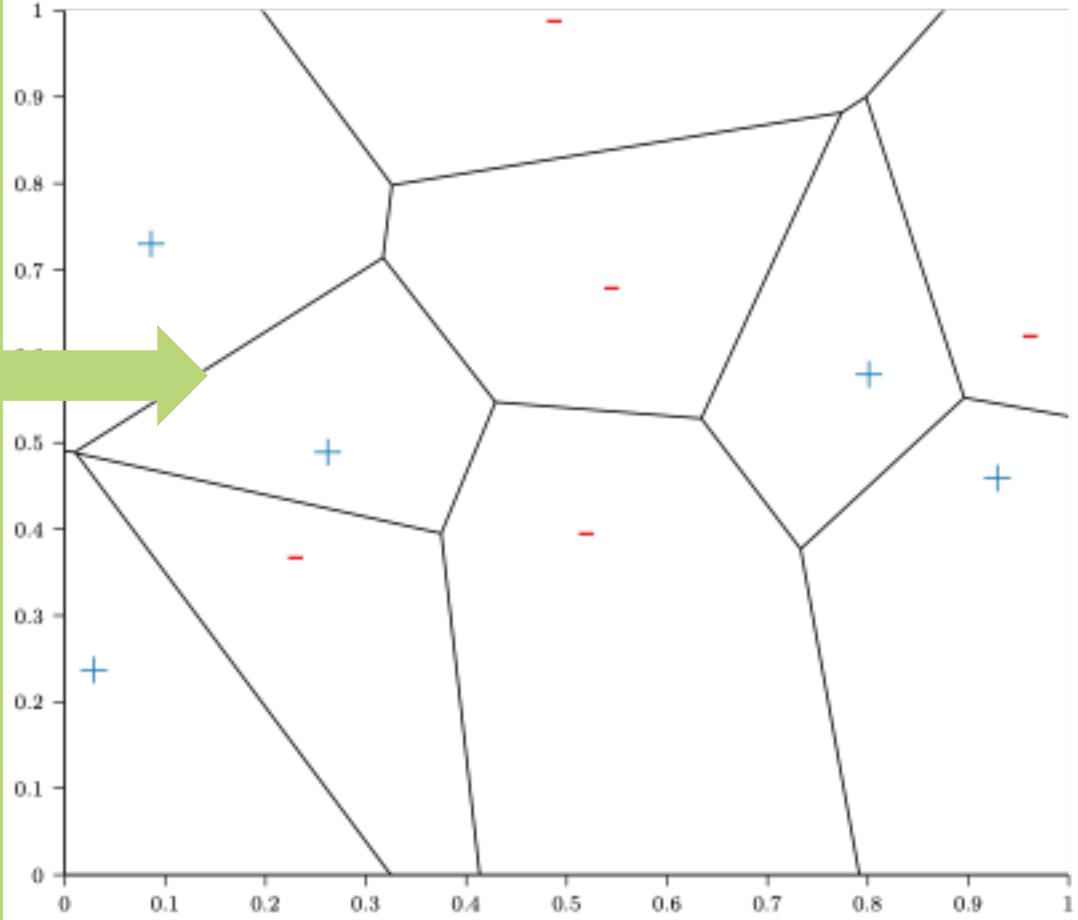
```
def train($\mathcal{D}$):
    Store $\mathcal{D}$

def h($\boldsymbol{x}'$):
    Let $\boldsymbol{x}^{(i)}$ = the point in $\mathcal{D}$ that is nearest to $\boldsymbol{x}'$
    return $y^{(i)}$
```

Nearest Neighbor: Example

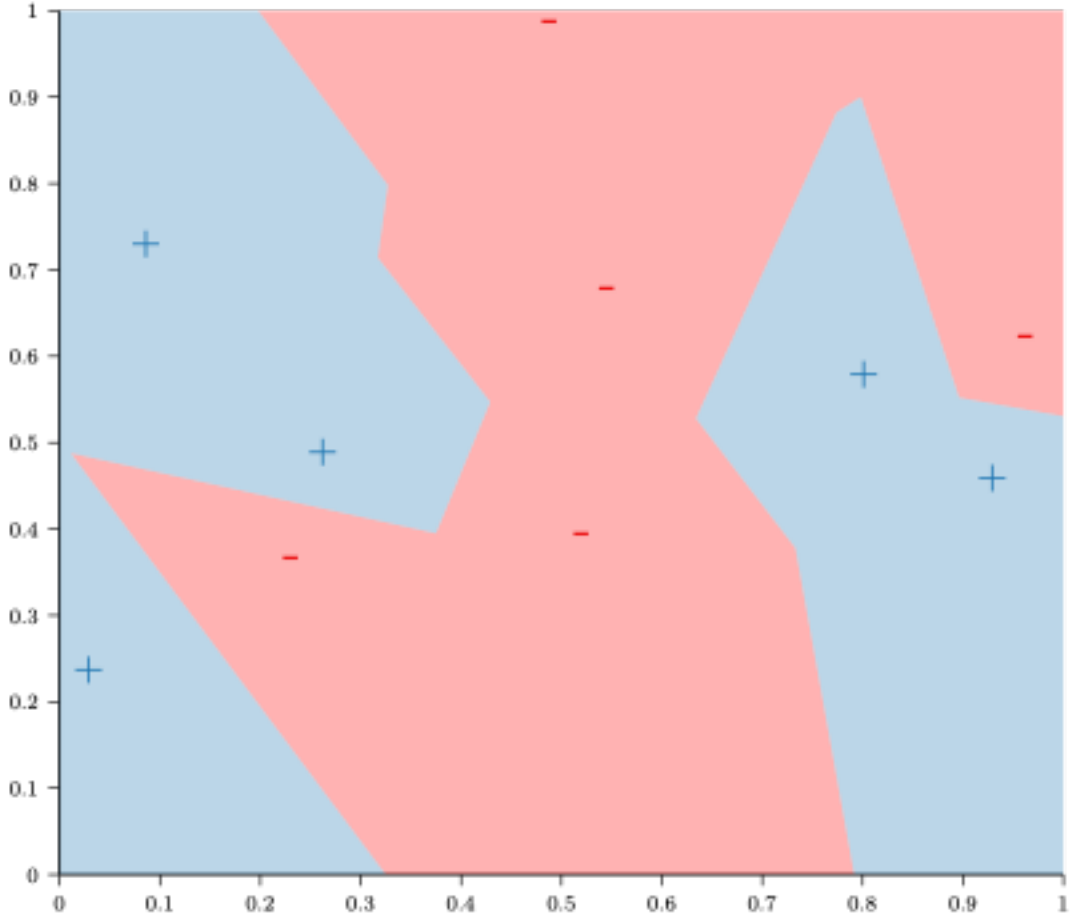


Nearest Neighbor: Example

- This is a **Voronoi diagram**
- Each **cell** contain one of our training examples
- **All points within a cell** are **closer** to that training example, than to any other training example
- **Points on the Voronoi line segments** are **equidistant** to one or more training examples



Nearest Neighbor: Example



The Nearest Neighbor Model

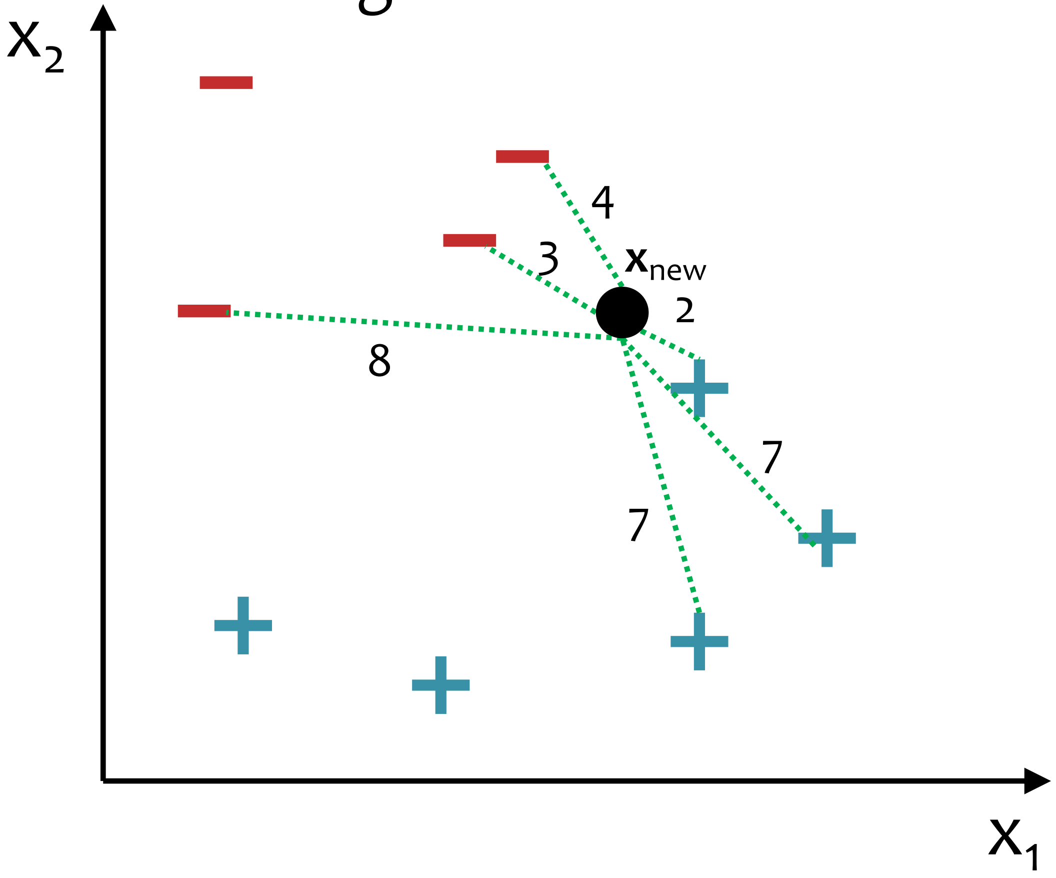
- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***

k-Nearest Neighbors: Algorithm

```
def set_hyperparameters(k, d):
    Store k
    Store d(·, ·)

def train($\mathcal{D}$):
    Store $\mathcal{D}$

def h($\boldsymbol{x}'$):
    Let S = the set of k points in $\mathcal{D}$ nearest to $\boldsymbol{x}'$
            according to distance function
            d(u, v)
    Let v = majority_vote(S)
    return v
```

k-Nearest Neighbors

Suppose we have the training dataset below.



How should we label the new point?

It depends on k:

if k=1, $h(\mathbf{x}_{new}) = +1$

if k=3, $h(\mathbf{x}_{new}) = -1$

if k=5, $h(\mathbf{x}_{new}) = +1$

KNN: Remarks

Distance Functions:

- KNN requires a **distance function**

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

- But there are other choices (e.g. **Manhattan distance**)

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{m=1}^{M} |u_m - v_m|$$

KNN: Computational Efficiency

- Suppose we have N training examples and each one has M features
- Computational complexity when k=1:

Task	Naive	k-d Tree
Train	O(1)	~ O(M N log N)
Predict (one test example)	O(MN)	~ O(2^M log N) on average

Problem: Very fast for small M, but very slow for large M

In practice: use stochastic approximations (very fast, and empirically often as good)

KNN: Theoretical Guarantees

Cover & Hart (1967)

Let h(x) be a Nearest Neighbor (k=1) binary classifier. As the number of training examples N goes to infinity…

$$\text{error}_{\text{true}}(h) < 2 \text{ x Bayes Error Rate}$$

"In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

very informally, Bayes Error Rate can be thought of as: *'the best you could possibly do'*

KNN: Remarks

In-Class Exercises

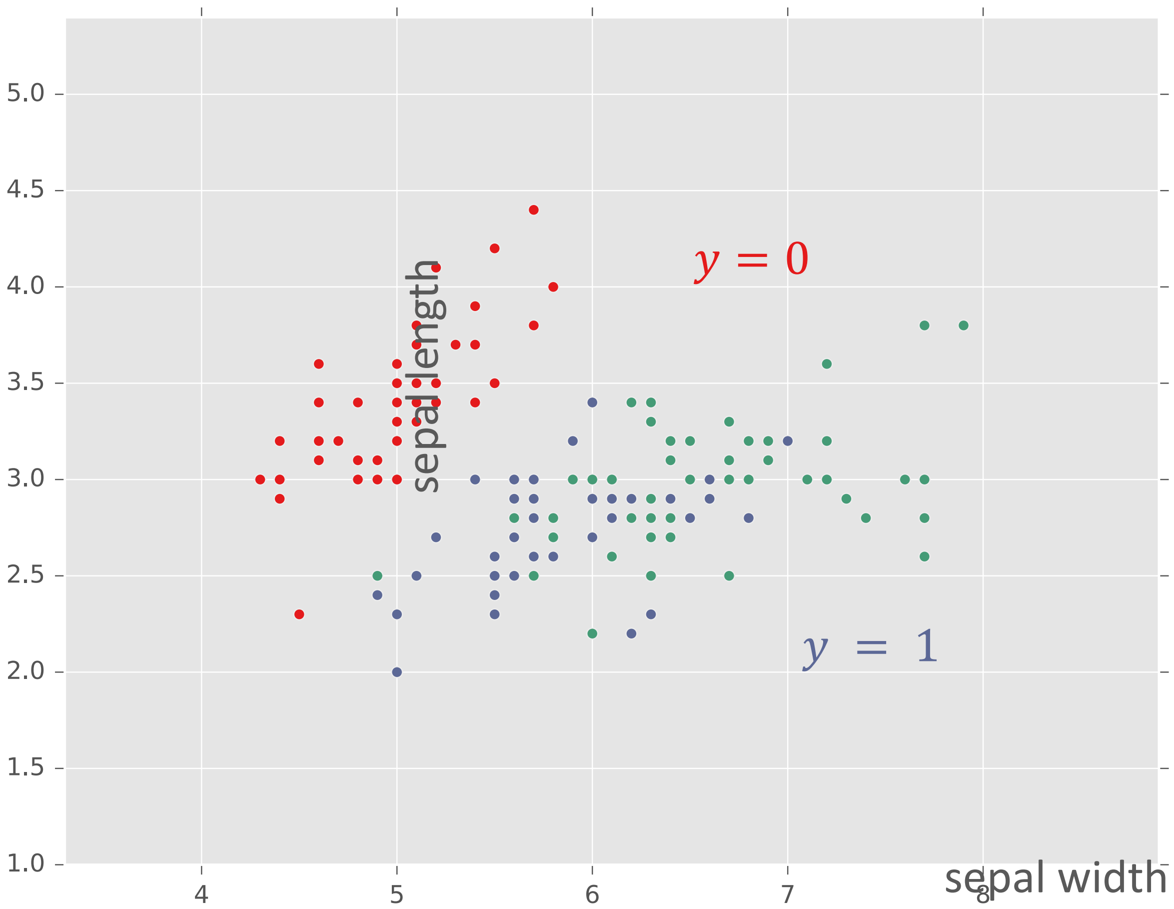
How can we handle ties for even values of k?

Answer(s) Here:

KNN: Inductive Bias

In-Class Exercise What is the inductive bias of KNN?

kNN on Fisher Iris Data



$y = 2$

Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

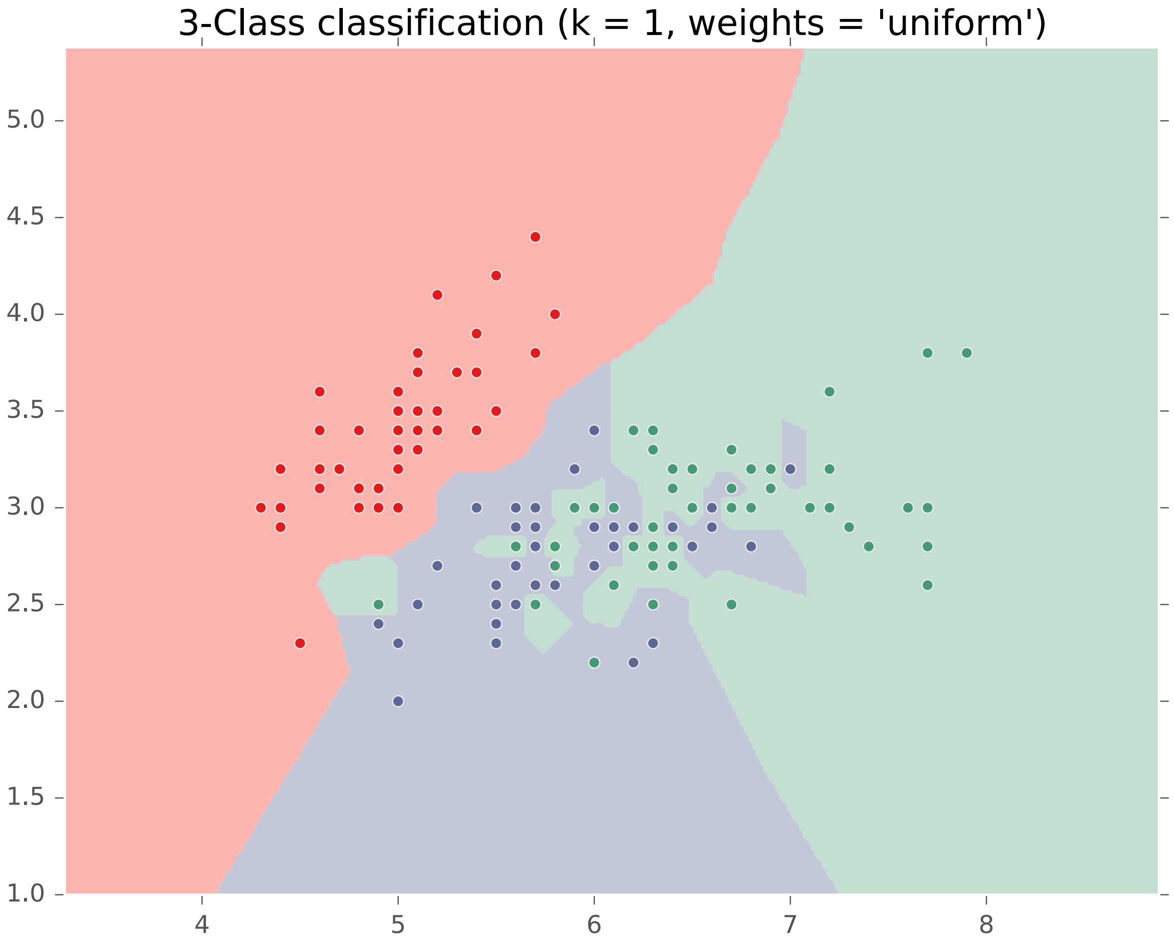


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

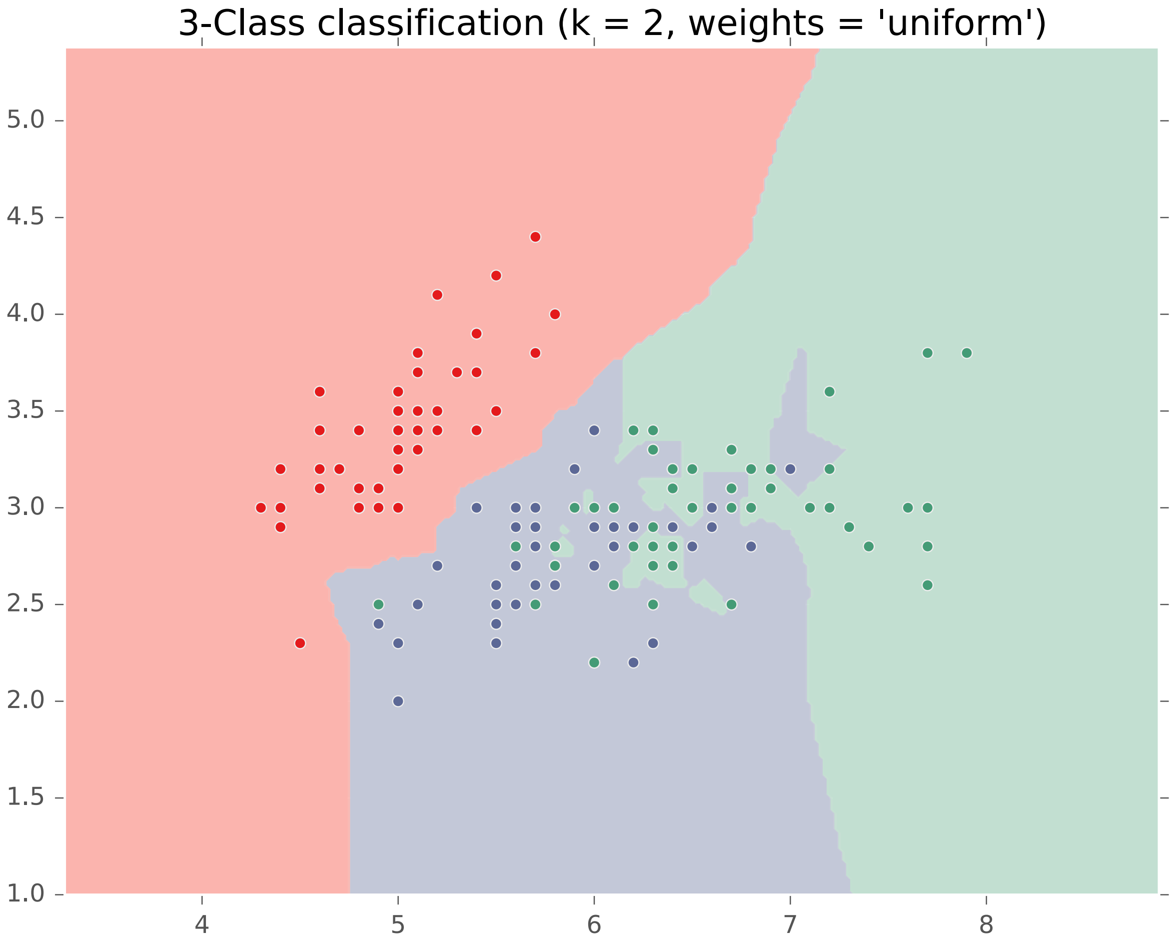


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

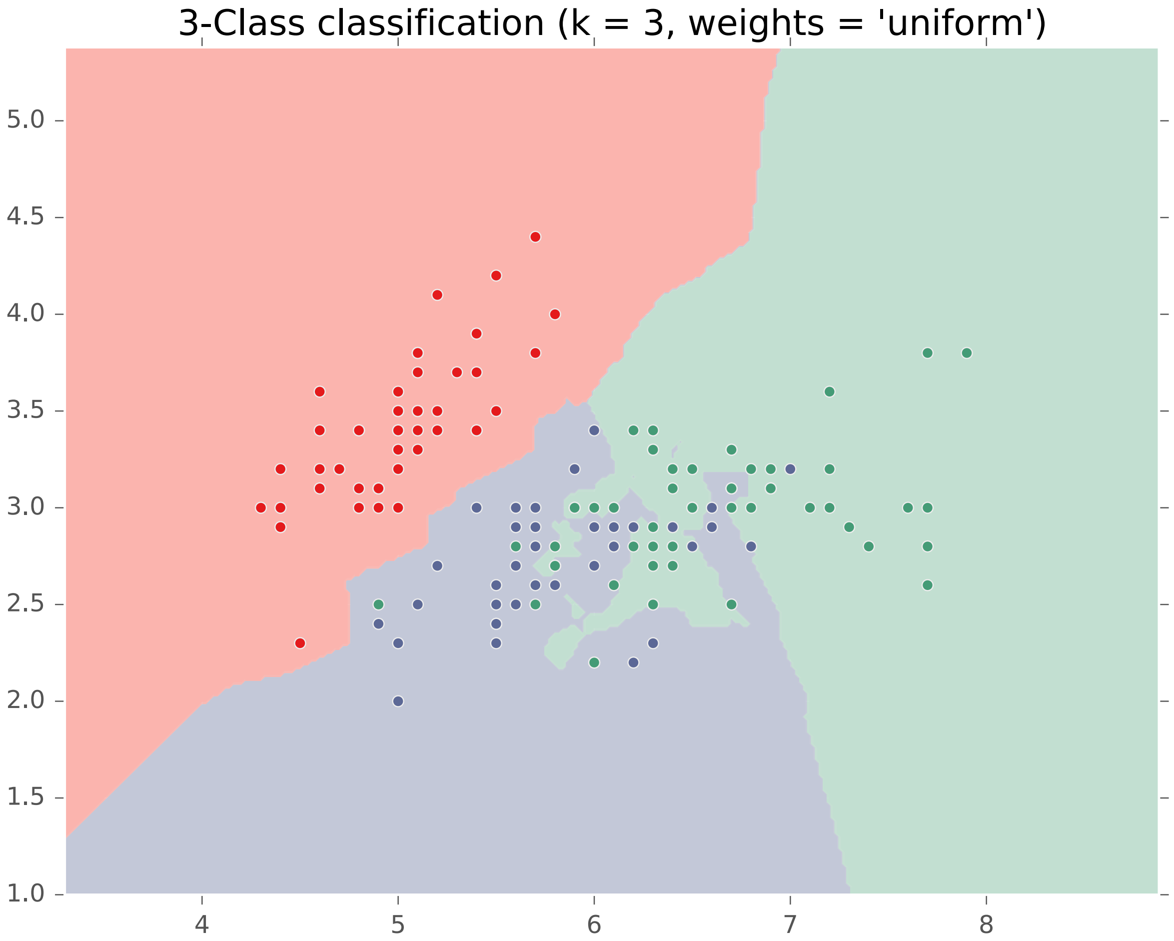


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

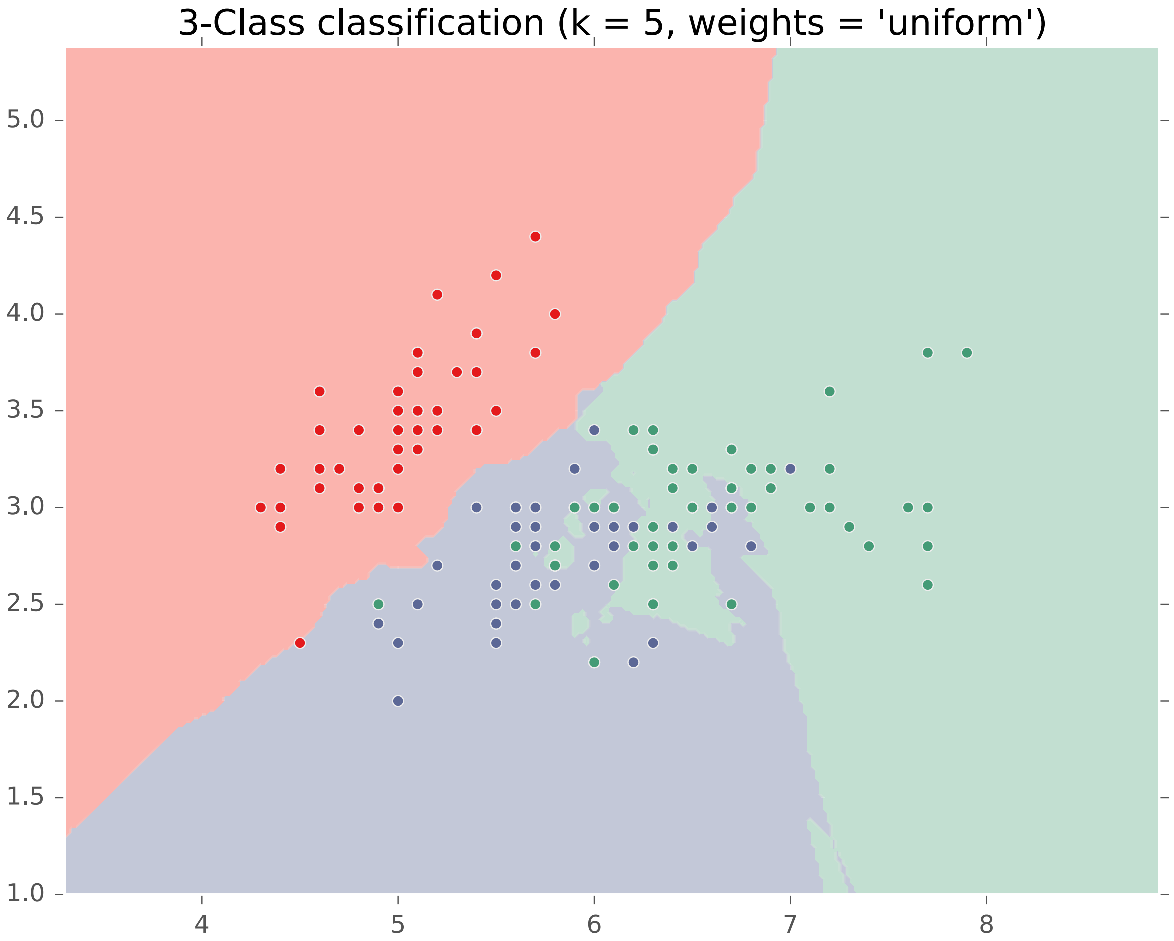


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

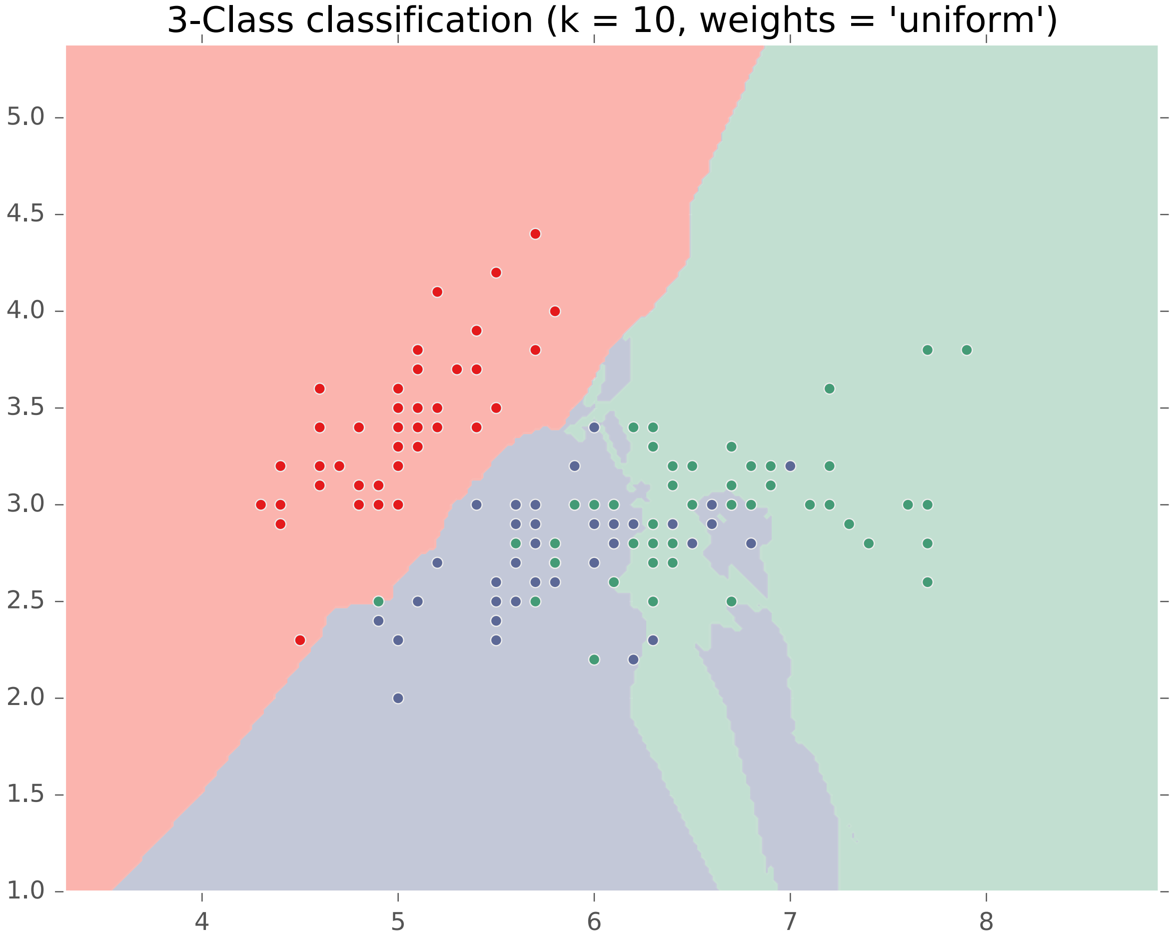


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

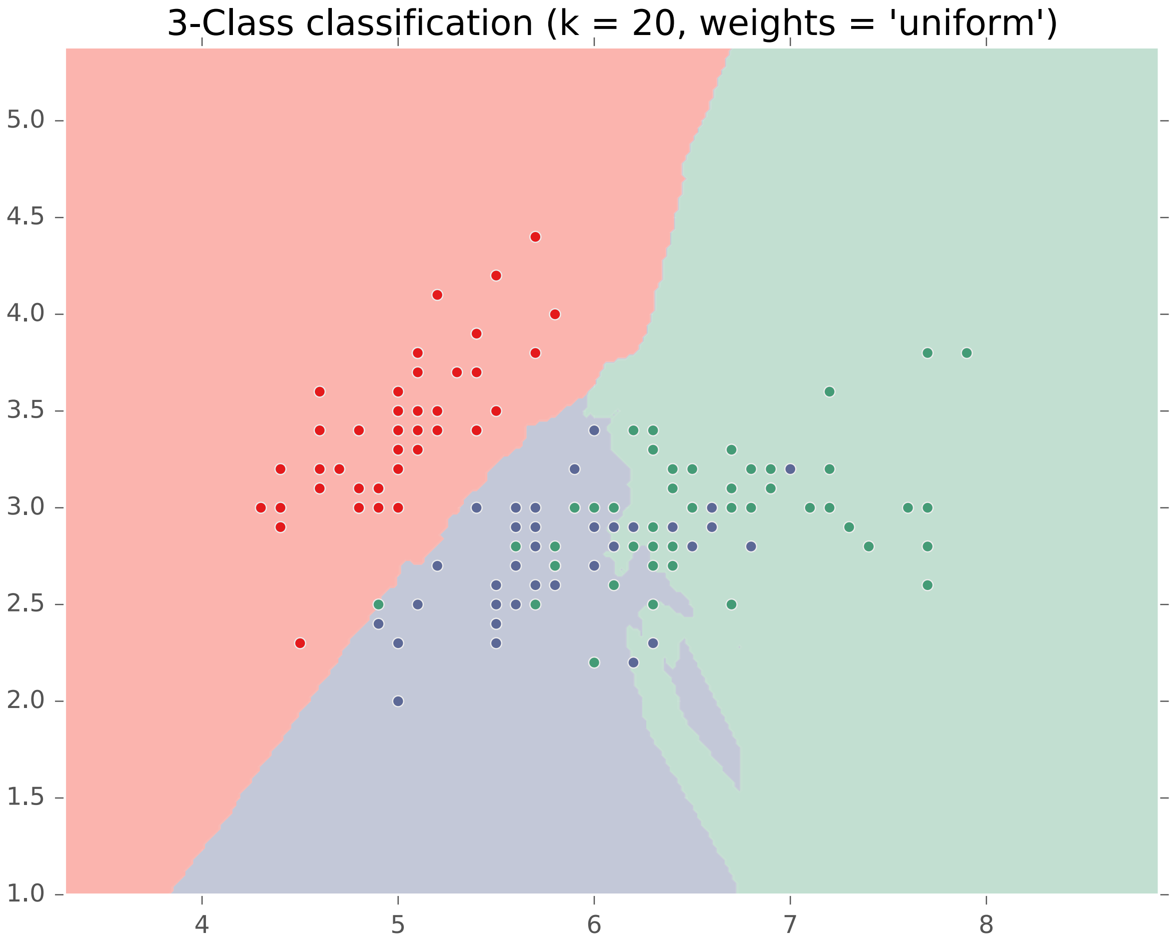


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

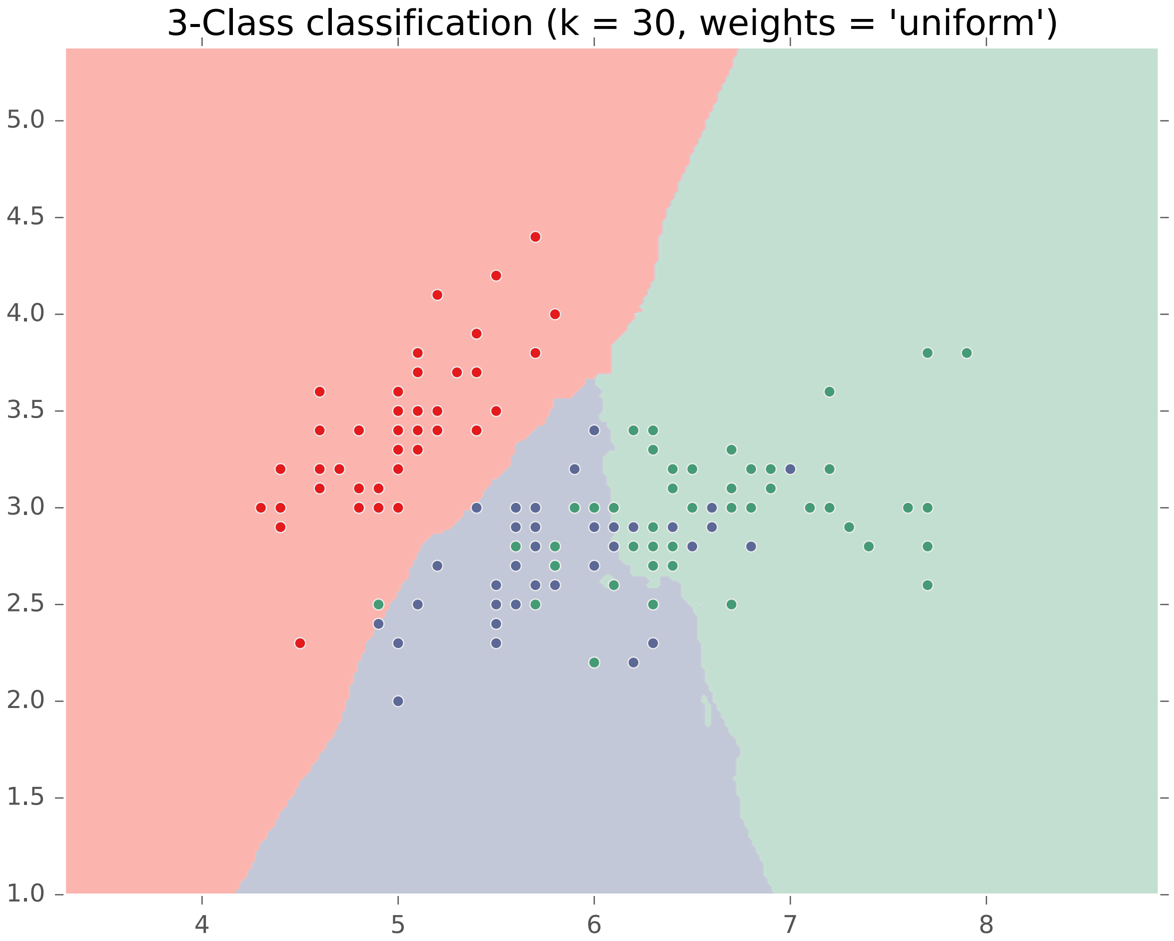


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

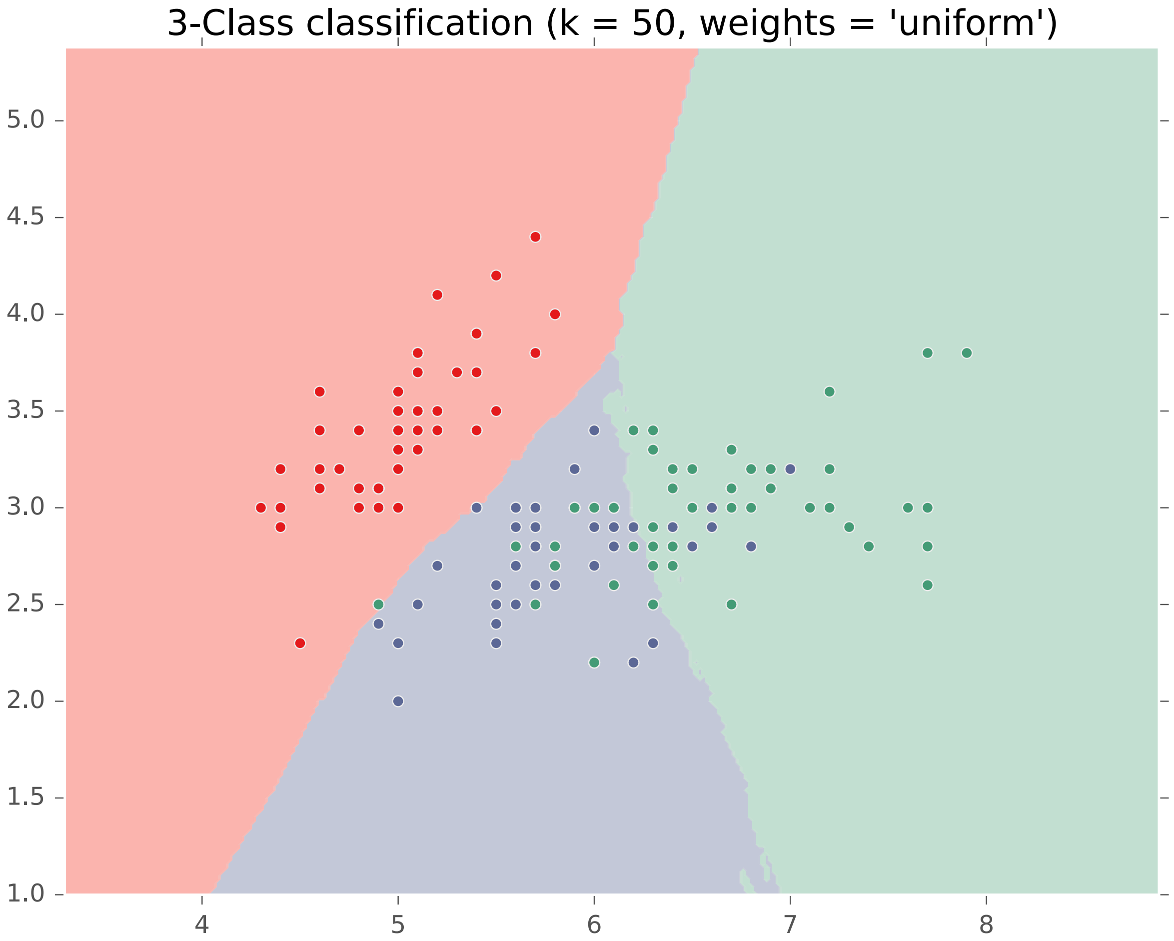


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

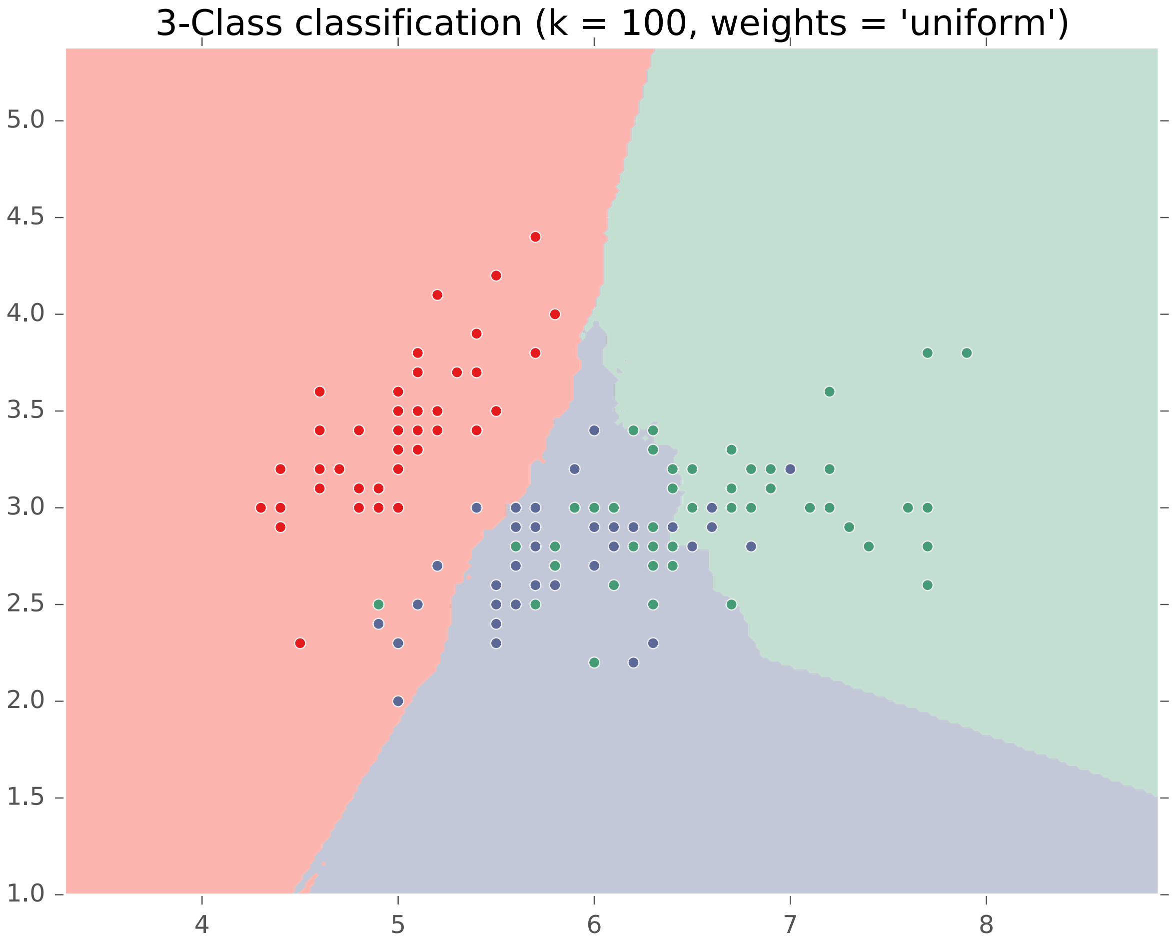


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

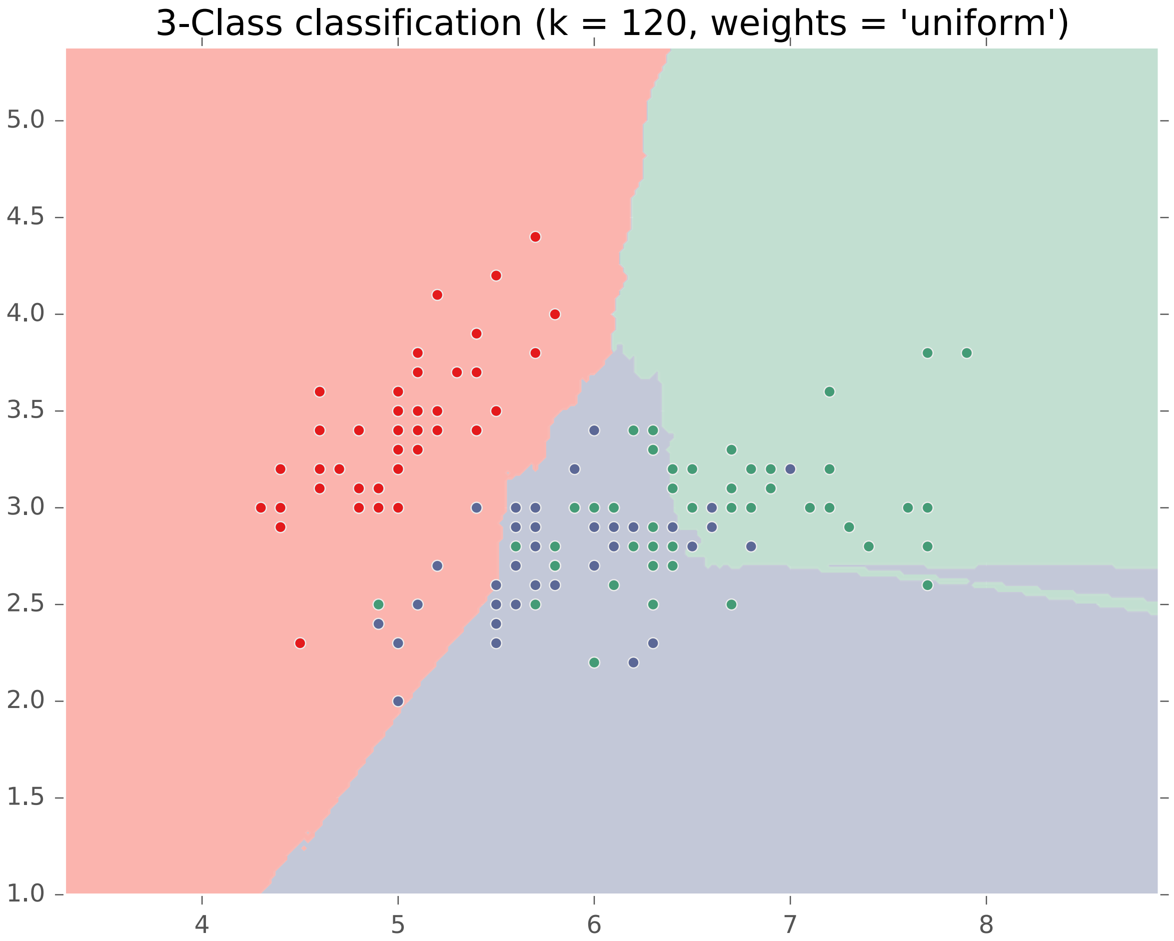


Figure courtesy of Matt Gormley

*k*NN on Fisher Iris Data

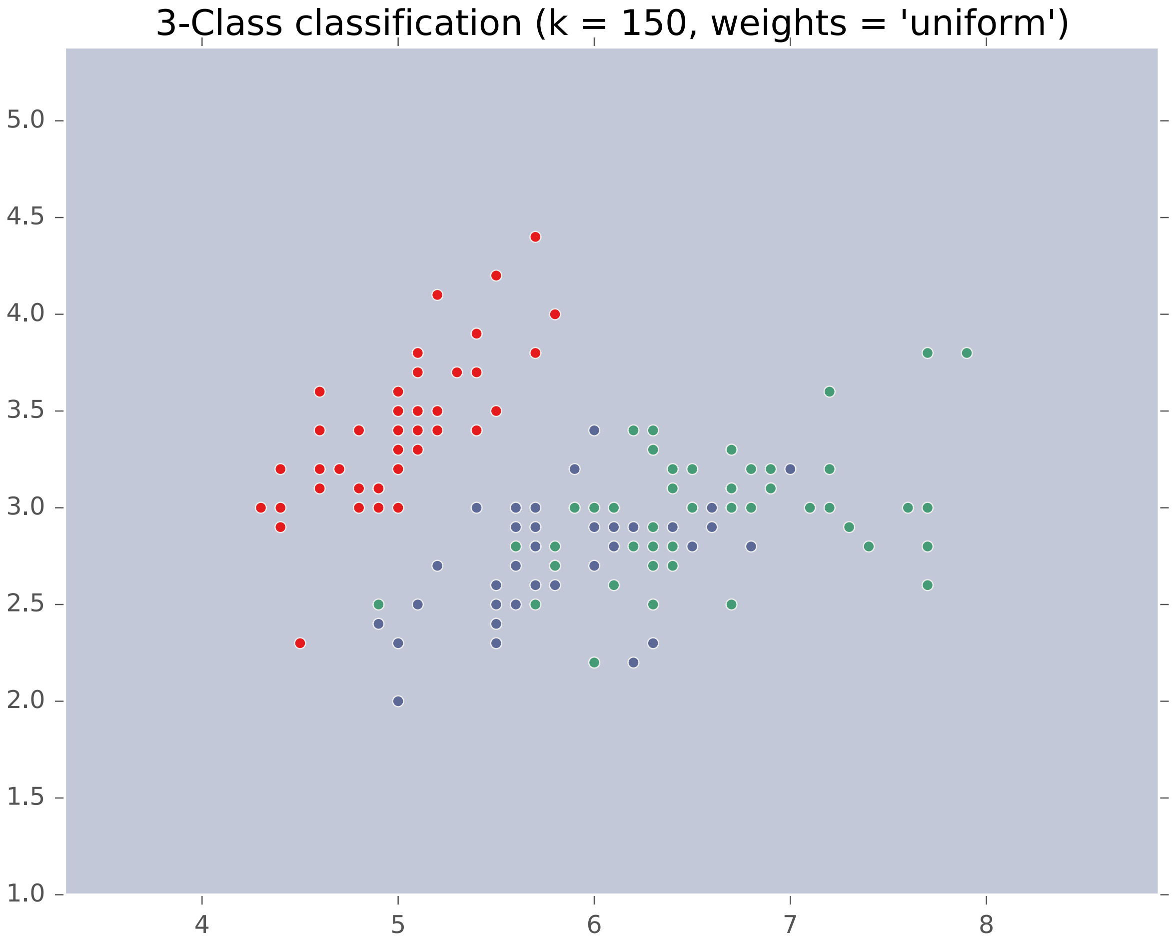


Figure courtesy of Matt Gormley

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: $O(ND + N\log(k))$
  - Affected by feature scale

KNN Learning Objectives

You should be able to…

- Describe a dataset as points in a high dimensional space [CIML]
- Implement k-Nearest Neighbors with O(N) prediction
- Describe the inductive bias of a k-NN classifier and relate it to feature scale [a la. CIML]
- Sketch the decision boundary for a learning algorithm (compare k-NN and DT)
- State Cover & Hart (1967)'s large sample analysis of a nearest neighbor classifier
- Invent "new" k-NN learning algorithms capable of dealing with even k

How on earth do we go about setting k?

You should be able to…

- Describe a dataset as points in a high dimensional space [CIML]
- Implement k-Nearest Neighbors with O(N) prediction
- Describe the inductive bias of a k-NN classifier and relate it to feature scale [a la. CIML]
- Sketch the decision boundary for a learning algorithm (compare k-NN and DT)
- State Cover & Hart (1967)'s large sample analysis of a nearest neighbor classifier
- Invent "new" k-NN learning algorithms capable of dealing with even k

How on earth do we go about setting k?

- This is effectively a question of model selection: every value of k corresponds to a different model.

- **WARNING:**
  - In some sense, our discussion of model selection is premature.
  - The models we have considered thus far are fairly simple.
  - In the real world, the models and the many decisions available to you will be much more complex than what we've seen so far.