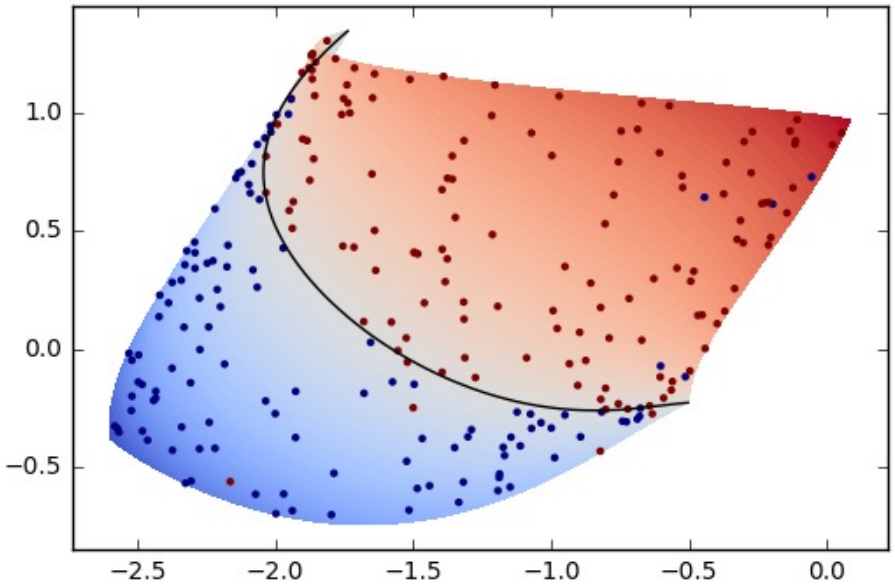
Lecture 8: k Nearest Neighbors



Gavin Kerrigan

Spring 2023

Some materials courtesy Padhraic Smyth, Alex Ihler.

Announcements

- HW1 grades available on Gradescope
  - Mean score: 92.4
- Regrade requests open for one week (until 4/26)
  - Submit via Gradescope
  - Only submit if we made a mistake while grading
  - We may re-grade your entire assignment, which can result in a higher or lower score
- HW2 posted on Canvas – due next Friday (4/28)
  - Start early
  - Post questions on Ed
- Discussion tomorrow:
  - Exploration of MNIST and Yelp datasets (HW2)

k Nearest Neighbors

Time-Space Complexity of Classifiers

The Curse of Dimensionality

Comparing Classifiers

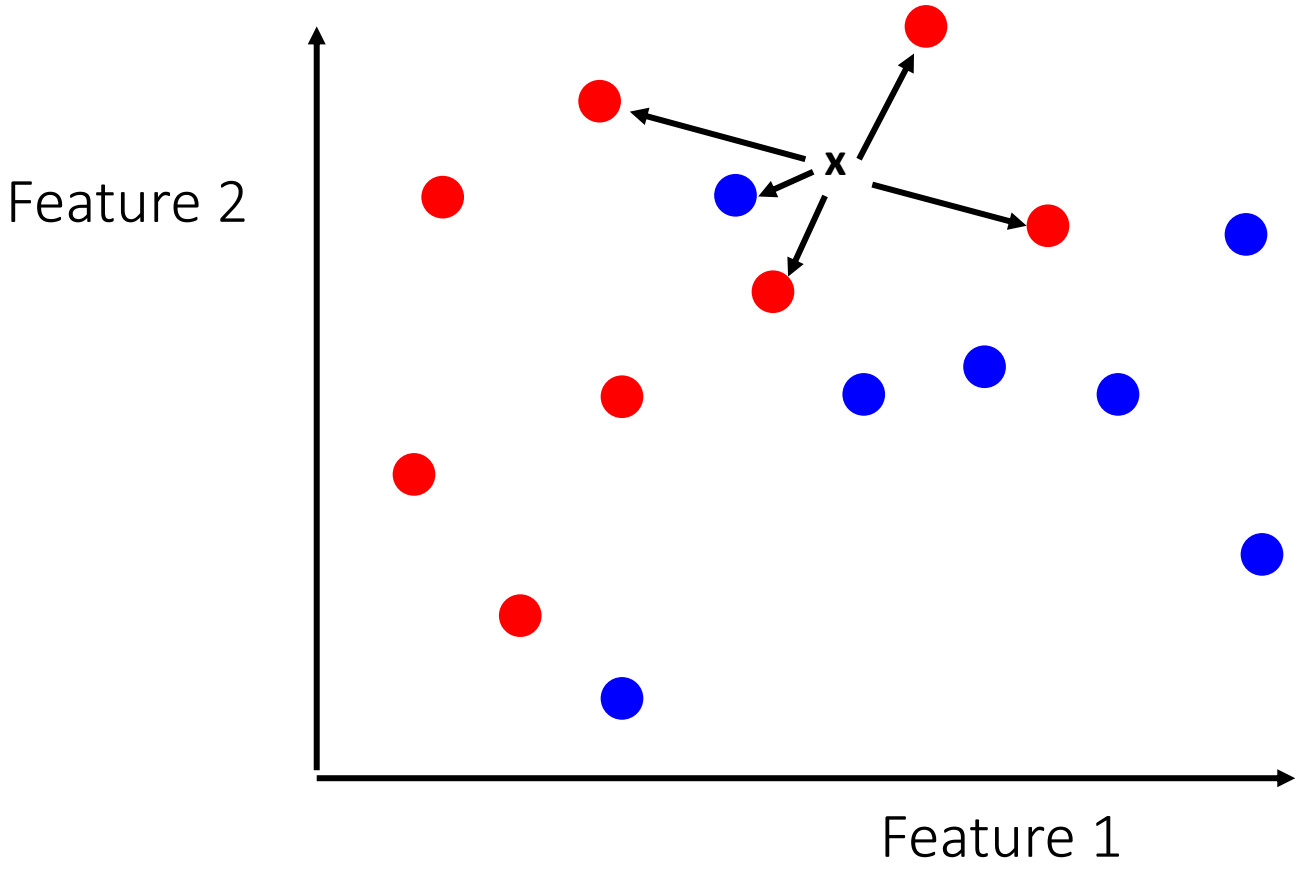
Review of kNN Classifier

Fix a number of neighbors k.

To make a prediction on an unlabeled datapoint $\boldsymbol{x}$:

1. Find the labels of the k nearest neighbors of $\boldsymbol{x}$ in the training data
   - i.e. evaluate $d(\boldsymbol{x}, \boldsymbol{x_i})$ for every feature vector $\boldsymbol{x}_i$ in the training data
   - Return the labels of the k datapoints with smallest distance to $\boldsymbol{x}$
2. Use the majority vote of the returned labels as the prediction for $\boldsymbol{x}$
   - e.g. if the label set is {1, 3, 3, 2, 1, 3}, predict label 3
   - Break ties randomly

Review of kNN Classifier

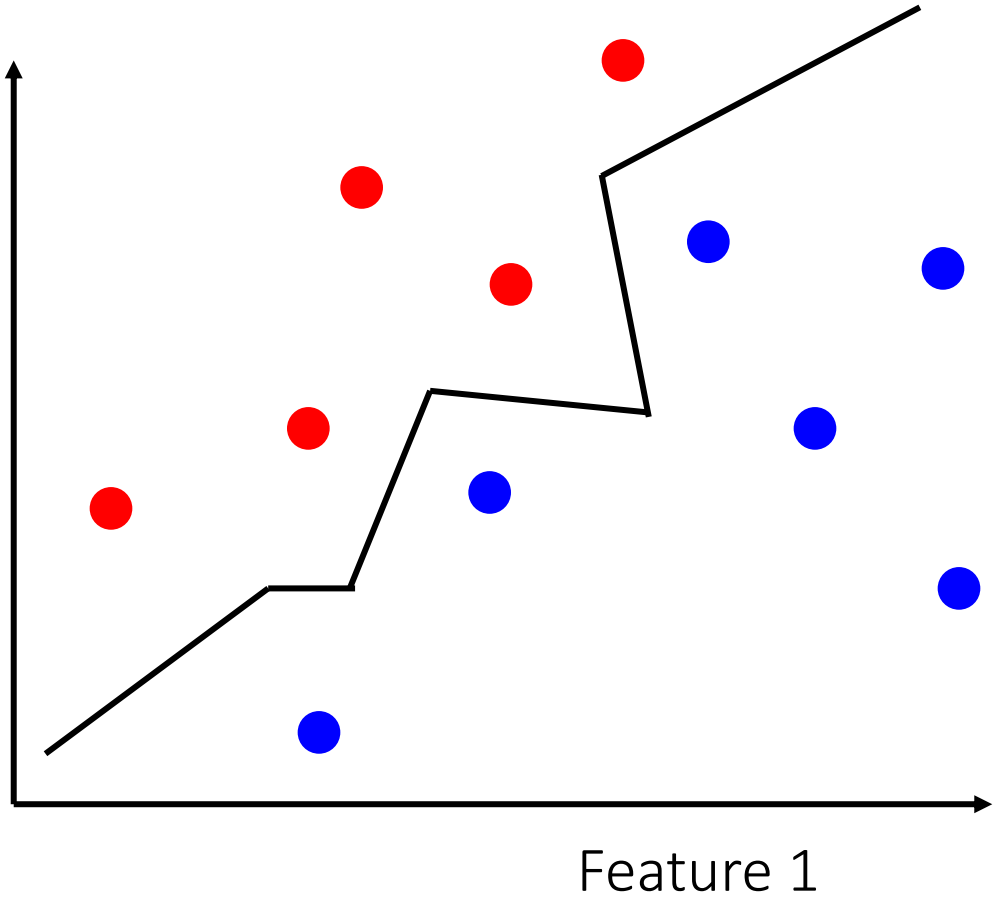


k = 1: predict blue

k = 3: predict red

k = 5: predict red

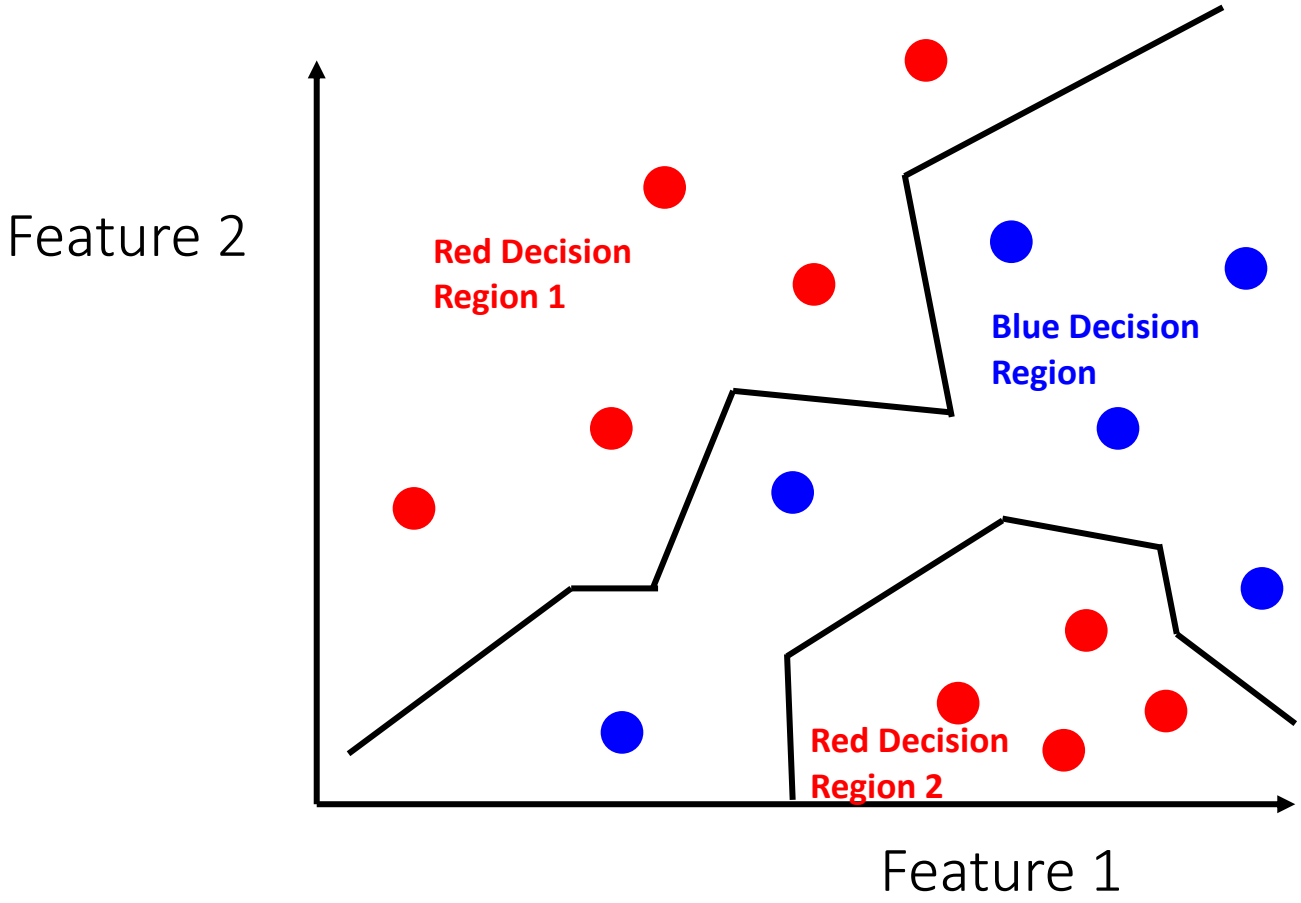
Decision Boundaries for kNN



kNN produces piecewise linear decision boundaries (shown here for k=1)

More complex than nearest-centroid

Decision Boundaries for kNN



kNN can also model disjoint decision regions

Here there are two separate disjoint parts of the input space that will be predicted as the red class

Nearest centroid can only produce one decision region per class

Wine Dataset

- 178 Italian wines, 13 features, 3 classes of wine

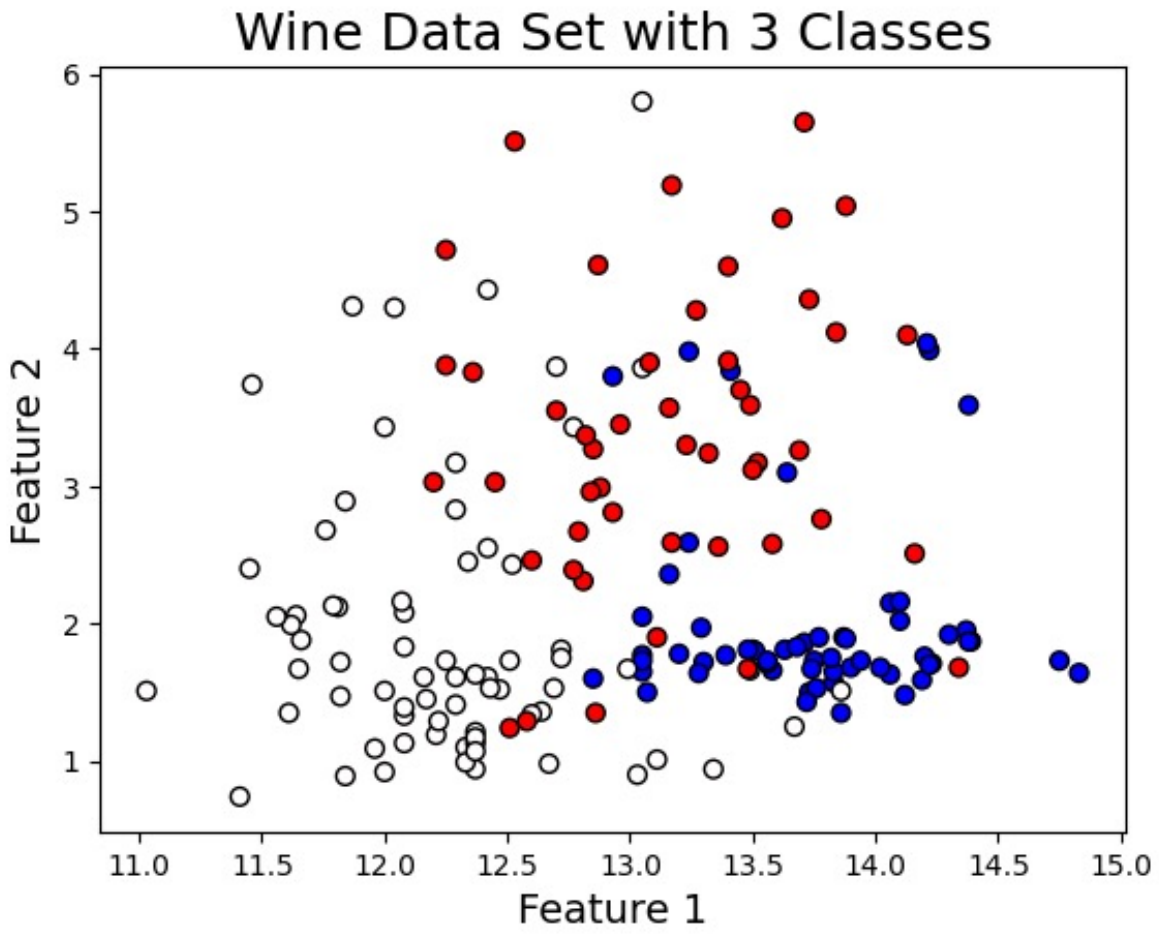
- Will use the first 2 features in the next few slides:
  - Feature 1 = alcohol content
  - Feature 2 = amount of malic acid

- In Python:

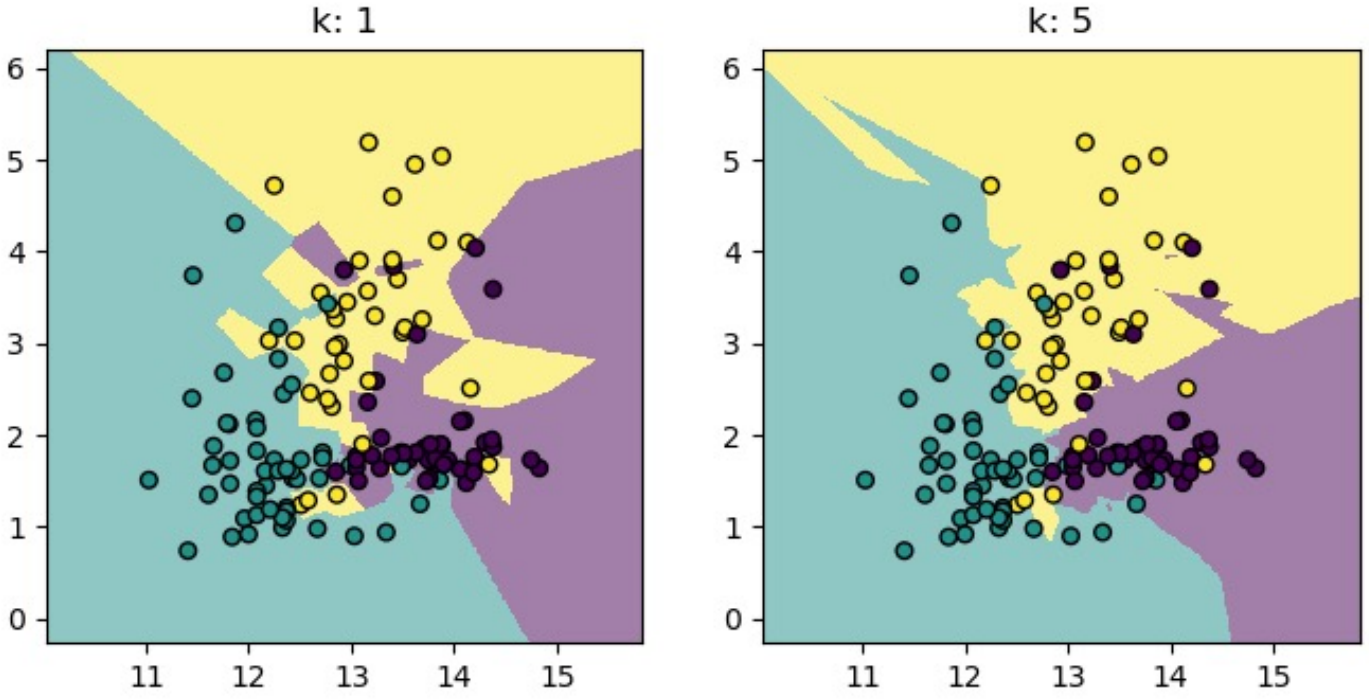
```
from sklearn.datasets import load_wine
wine_X, wine_y = load_wine(return_X_y=True)
```

Also, seed = 1234 as in Homework 1/2, and using 500 x 500 points for the grid for decision boundaries

Wine Data with First Two Features

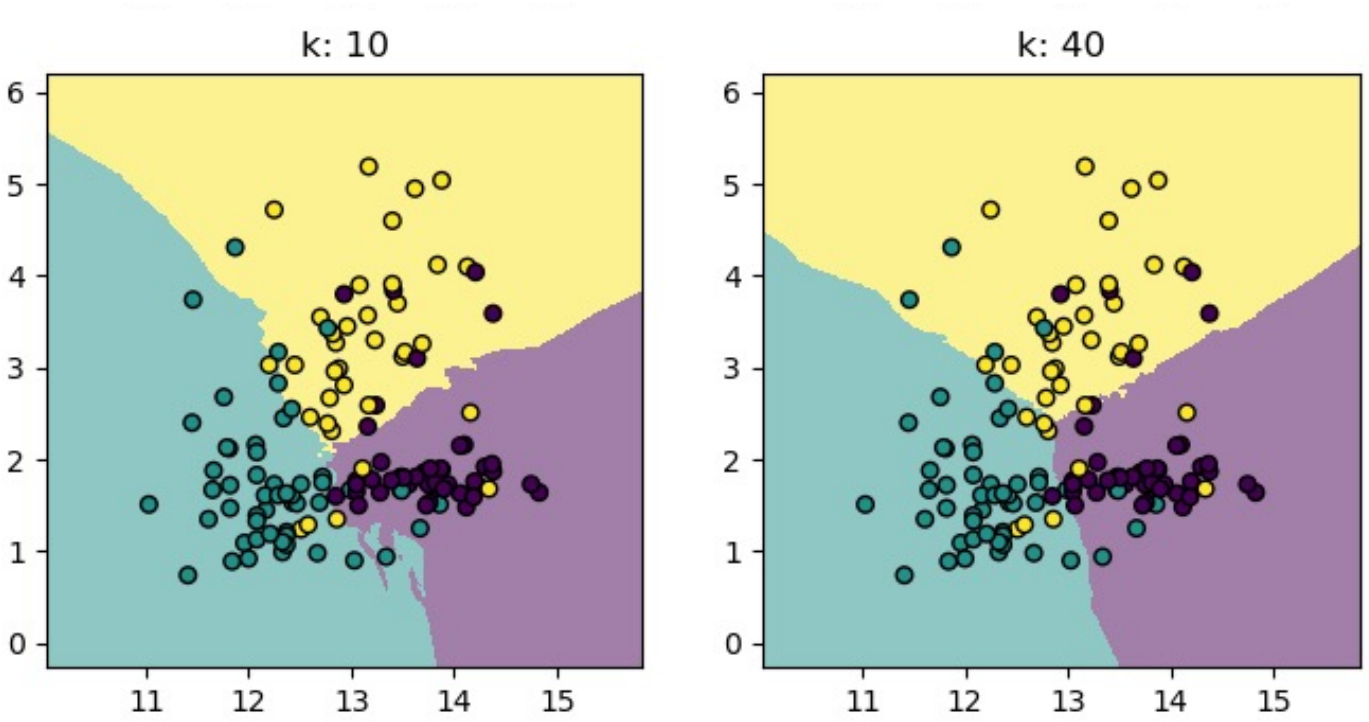


kNN Decision Boundaries (small k)



Decision regions can be complex for small k
(in general, will be *piecewise linear* for all values of k)

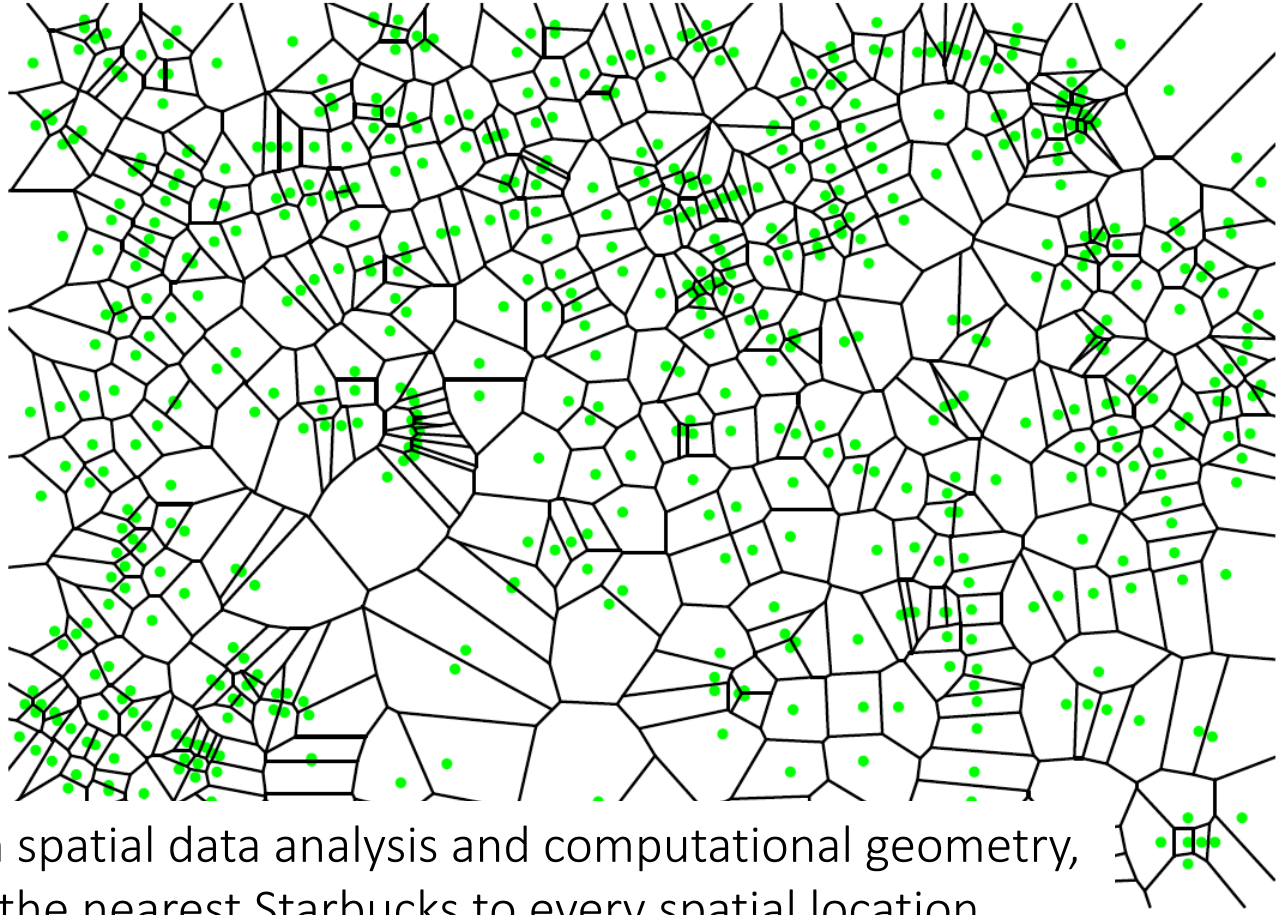
kNN Decision Boundaries (larger k)



More compact decision regions and smoother decision boundaries as k increases

Voronoi Diagrams

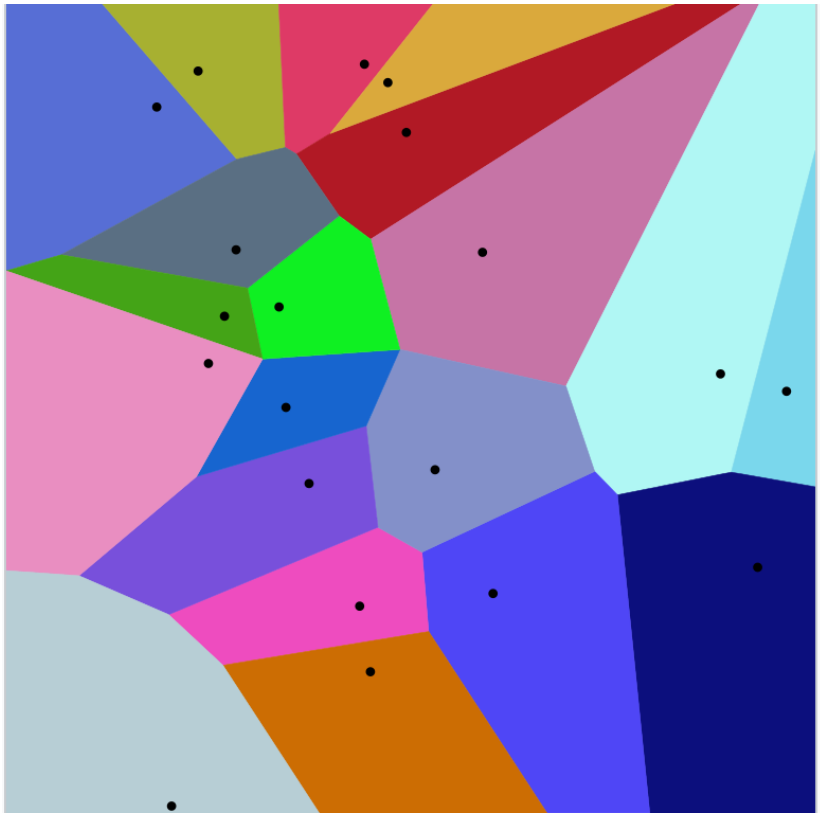
Georgy Voronoy



Widely used in spatial data analysis and computational geometry, e.g., where is the nearest Starbucks to every spatial location

Voronoi Diagrams

Choice of distance function has a strong effect on the decision regions:



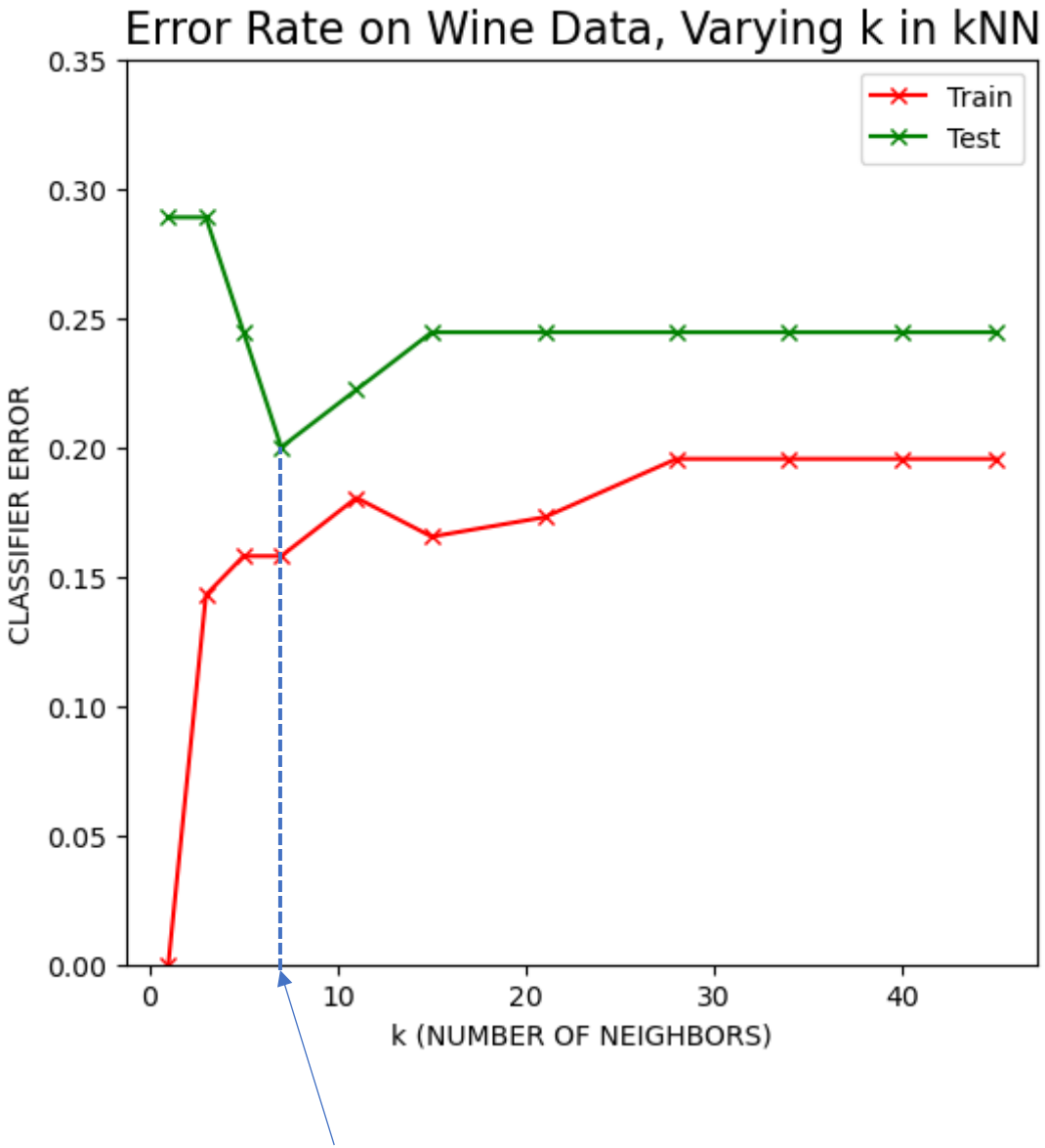
L2 (Euclidean) distance

L1 distance

https://en.wikipedia.org/wiki/Voronoi_diagram

How do we select k?

- Classification accuracy will typically vary with k
- Intuitively:
  - If k is too small, predictions can be too noisy
  - If k is too large, predictions based on points too far away from x
  - There is usually a happy medium where error is minimized
- k is a hyperparameter of the kNN method
- We can use a validation set to help tune the value of k



Best value of k (empirically)

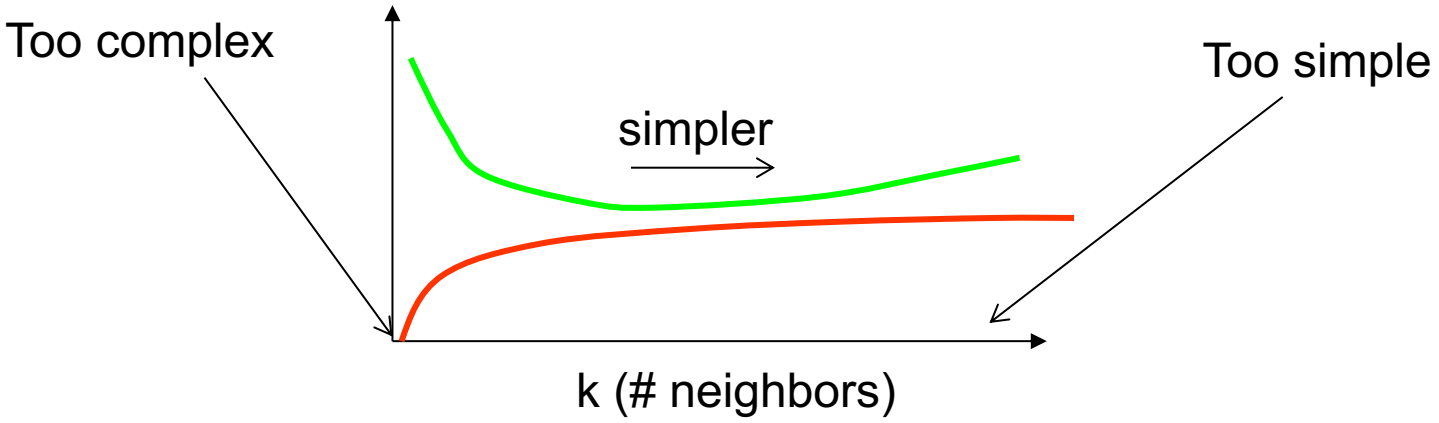
Note:

1. Test Error > Training Error
2. Test Error varies with k

Results generated using
75% train, 25% test

Complexity and Overfitting

- In general a complex classifier will predict all training points well….but might overfit and not generalize well to new data.
  - k = 1 : perfect memorization of examples (complex)
  - k = n: always predict majority class in dataset (simple)
  - 1 < k << n: may produce the best results (will vary by dataset)



This type of tradeoff (error and complexity) is common in ML

MNIST Data

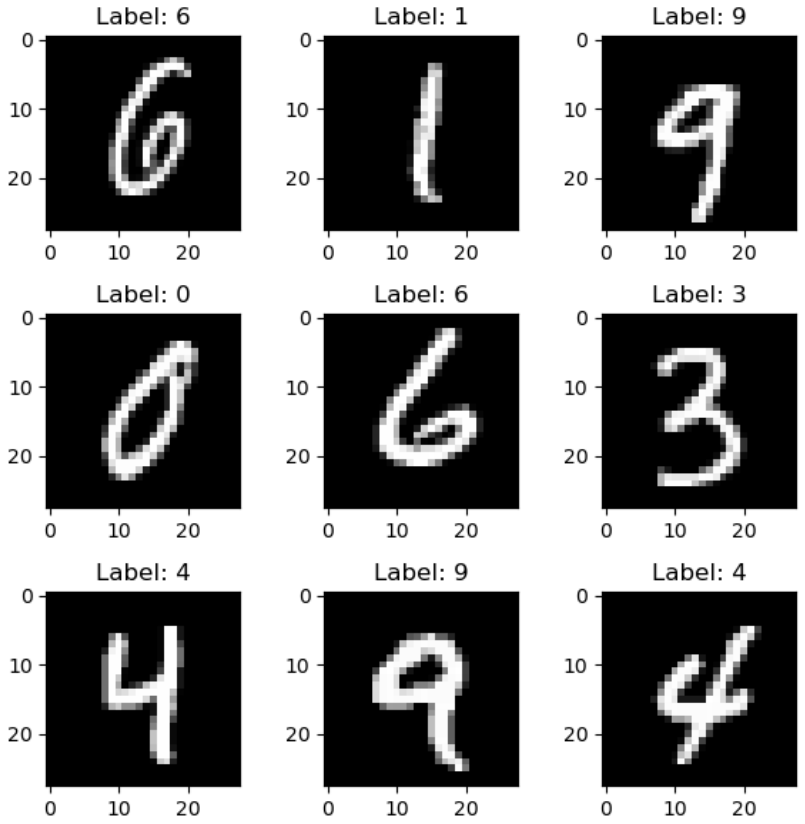
Classic dataset in ML

70,000 handwritten digits,
28 x 28 gray-scale pixels
Feature vector dim d = 28 x 28 = 784

10 class labels y ∈ {0, 1, 2,…. 9}

Used in Homework 1 (with 75% for train)

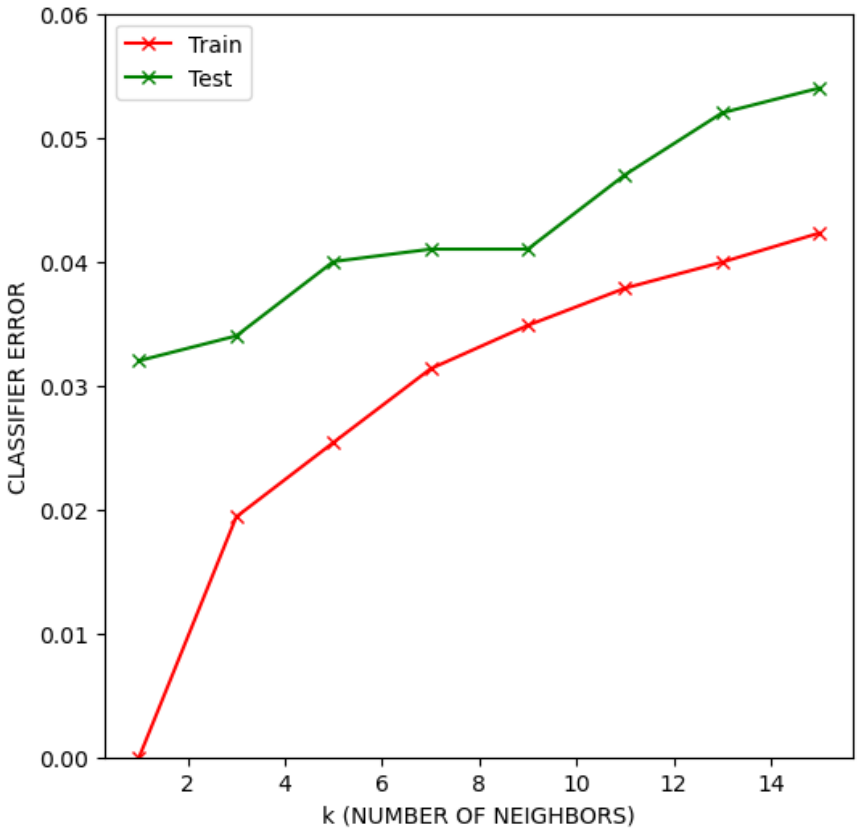
Data originally created by
Yann LeCun (Turing Award winner)
in 1998 for ML research.
See: Le Cun, Bottou, Bengio, Haffner,
Proceedings of IEEE, 1998



Error Rates for MNIST with kNN

- Accuracy of kNN on MNIST for different values of k
- Used 10,000 digits for training, and 1000 for test (to save computation time)
- What is a "good" error rate?
- Baseline error rate = 90%
  - All classes occur equally frequently, i.e., frequency 1/10
  - So, predicting a single class all the time, or a random class, will be correct only 10% of the time

Error Rates for MNIST Classification with kNN



Error rate is much lower than with nearest centroid (~20%)

On this data, k=1 is best (but will vary depending on dataset, training size, etc)

Distances are Sensitive to Dimensionality

Nearest-Centroid and kNN rely on distances

Distances can be unreliable in high-dimensional spaces:
- poorly scaled features can dominate distance
- distance may be sensitive to noisy irrelevant features that are not useful in predicting the class label

Methods like nearest-centroid and kNN are unable to select the relevant features and ignore the rest, and so can easily overfit if there are many noisy irrelevant features

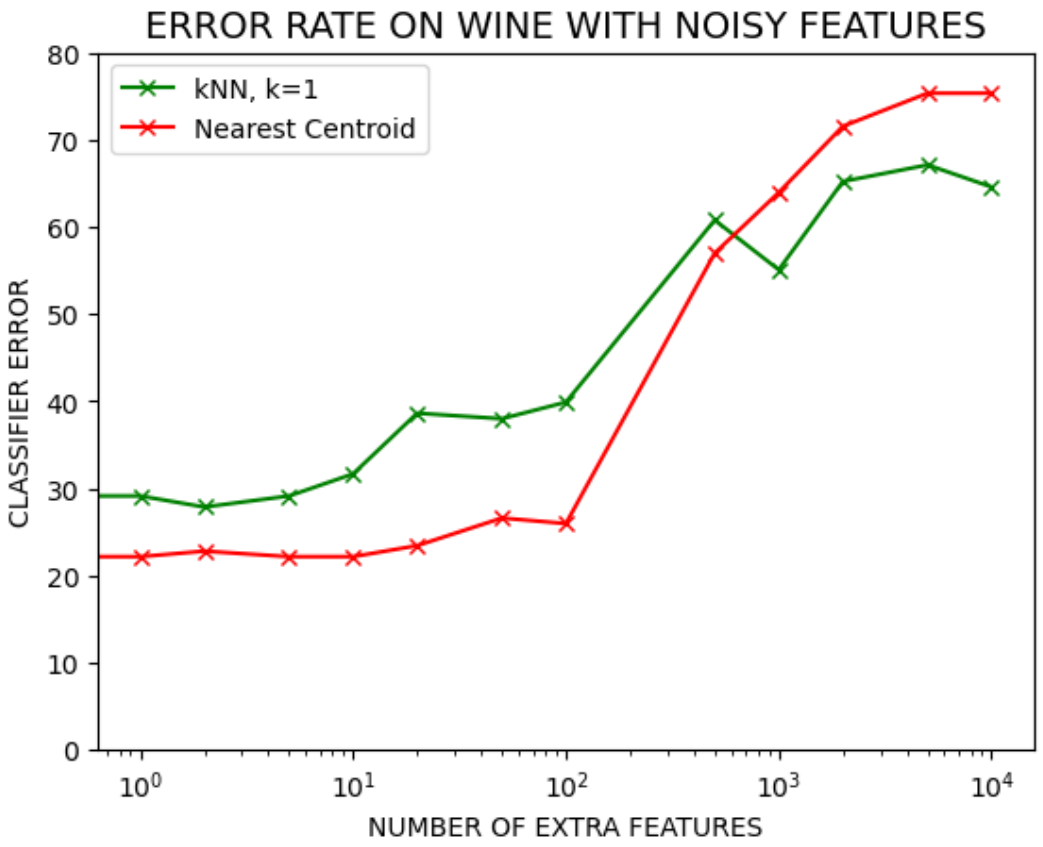
(Other classifiers such as decision trees (later in the quarter) are less sensitive to dimensionality)

Effect of adding Noisy Features (Wine)

Adding noisy features causes an increase in error rate

Each added feature is random noise, each feature on a scale of [0,1]

nTrain = 20
nTest = 158
k = 1

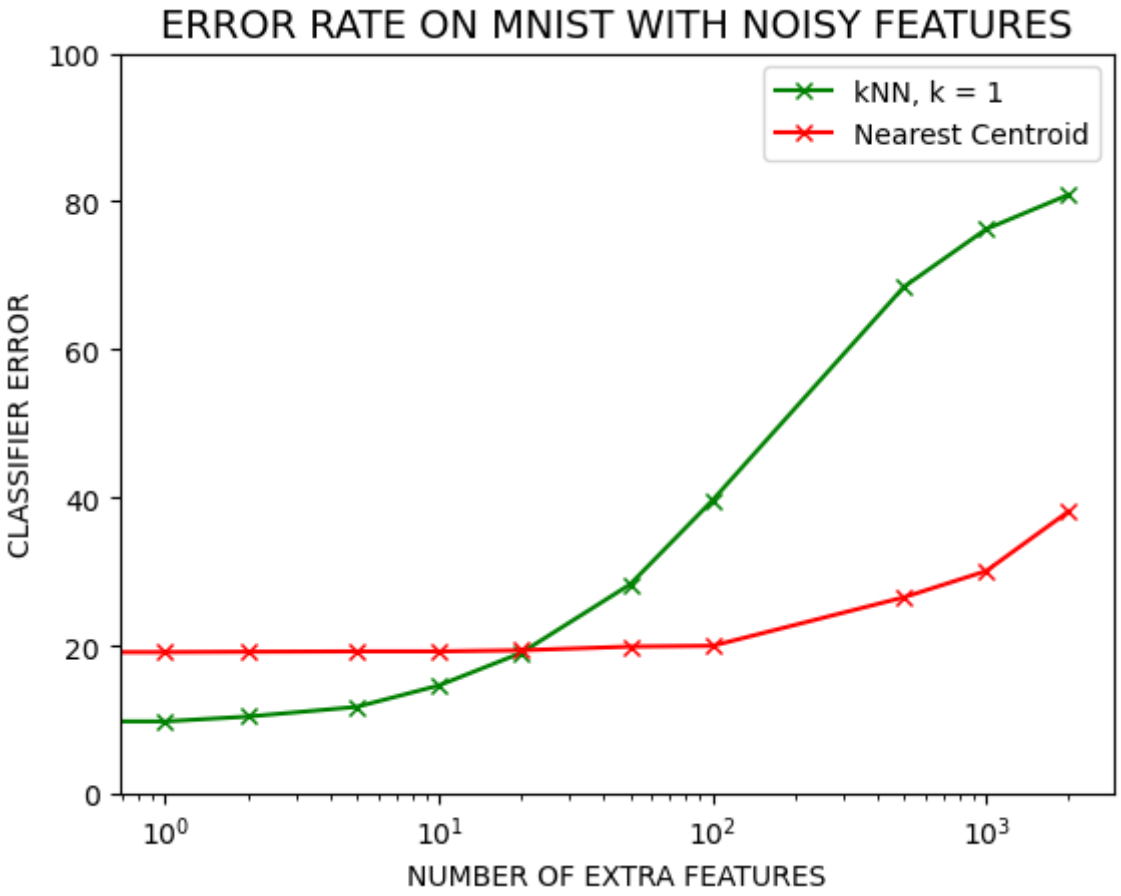


Effect of adding Noisy Features (MNIST)

Adding noisy features causes an increase in error rate

Each added feature is random noise, each feature on a scale of [0,1000]

Ntrain = 2000
k = 1



Summary of kNN Classifier

- Basic properties
  - Small k: complex boundaries
  - Larger k: smoother boundaries
- Strengths
  - Simple to implement
  - Only one hyperparameter to tune
  - can be quite accurate on some problems
- Weaknesses
  - Requires storage of all training data for predictions
    -> not ideal from a memory or speed perspective
  - Can lose accuracy as dimensionality d increases

Questions?

k Nearest Neighbors

Time-Space Complexity of Classifiers

The Curse of Dimensionality

Comparing Classifiers

Time and Space Complexity of Classifiers

- Variables:
  - n = number of examples
  - d = dimensionality of each feature vector
  - C = number of classes
  - (for kNN, k = number of neighbors)
  - We will assume that n >> k and n >> C

- "Big O" notation: how an algorithm scales in the worst case
  - E.g., $O(n\,d)$ => linear in n and in d
  - E.g., $O(d\,n^2 + C\,n)$ => quadratic in n

- We are interested in
  - Both time and space complexity
  - For both training and prediction phases of classification

Why do we care about Complexity?

- Training phase
  - In real applications n and d (and sometimes C) can be very large
    - E.g., n = millions, d = 100,000, etc
    - Ideally want to avoid super-linear scaling, e.g., $O(n^2)$, $O(d^3)$, etc
    - But $O(n\,d)$ is fine – and quite common for ML algorithms
  - Time-complexity
    - May need to train 1000's of models to evaluate hyperparameters
    - Or may want to retrain models regularly (e.g., multiple times per day)
  - Space-complexity
    - May need to store data on memory-limited GPUs

Why do we care about Complexity?

- Prediction phase
  - Time-complexity (per individual prediction)
    - we may want it to be very fast (< milliseconds)
    - e.g., real-time user interactions on a mobile device or Web interface
  - Space-complexity
    - Storing a large model (e.g., billions of weights) might not be practical on a mobile device (memory and battery limitations)

Time-Space Complexity of Nearest-Centroid

- Training the classifier
  - Time: $O(n\ d)$
    - C sums, each addition is O(d), total of elements summed is n
  - Space: $O(n\ d + n + C\ d)$

    ….. Data matrix + labels + C centroids

    = $O(n\ d + C\ d)$ which is effectively $O(n\ d)$ since usually $n >> C$

- Prediction per **x** , with the classifier
  - Time: $O(C\ d + C) = O(C\ d)$
    - Compute distance of **x** to C centroids: O(d) per distance, C times
    - Plus finding the minimum distance
  - Space: $O(C\ d)$
    - Store C centroids, each of dimension d

Time-Space Complexity of kNN

- Training?
  - Effectively no training with kNN

- Space complexity of Prediction per **x**
  - Space: $O(nd + n) = O(nd)$, to store the training data

- Time complexity of Prediction per **x**
  - We need $\text{dist}(\mathbf{x}, \mathbf{x}_i)$ for all training points x_i, $i = 1,\ldots n$
  - This is $O(nd)$
    ...each distance computation is $O(d)$, we do this n times

  - Next: finding the k smallest distances, see next slide

Time Complexity of Prediction for kNN (ctd)

- Now need to find the k smallest distances
  - Option 1: Could sort all n distances, this would add $O(n \log n)$
  - Option 2: we can do much better…
    - as we compute the n distances (sequentially) we could keep a sorted list of the k smallest distances we have seen so far
    - Check if current distance is smaller than any of the k distances
    - If so, insert it and delete the largest of the k distances
    - This is $O(k)$ every time we compute a distance, so $O(nk)$ in total
  - Option 3: we can be cleverer and use a heap data structure
    - Heap: data structure (e.g., binary tree) that allows insert/delete with a sorted list of length k in time $O(\log k)$, instead of $O(k)$
    - Idea: avoid full scan of k items -> search via a tree-structured list

Time Complexity of Prediction for kNN (ctd)

Option 1: $O(n\,d + n \log n)$

Option 2: $O(n\,d + n\,k)$

Option 3: $O(n\,d + n \log k)$

Comments

- Options 2 or 3 will generally be faster than Option 1 since typically $\log n$ will be larger than k
- For small k, the difference between Options 2 and 3 might not be significant, $n\,k$ versus $n \log k$
- If d is much larger than k, then $O(n\,d)$ will dominate
- Other options are also possible, depending on n, k, d, tradeoffs

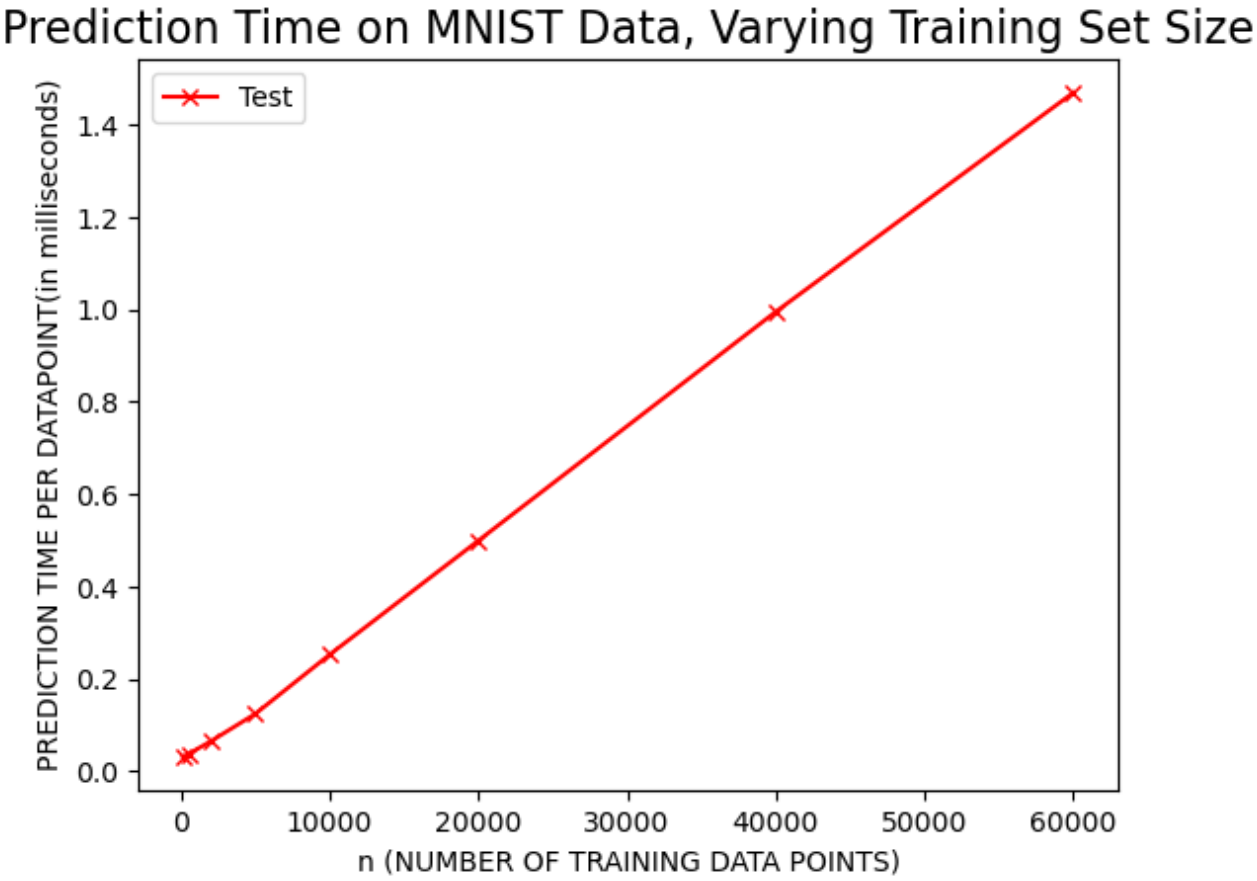
- Note that we ignore the cost of doing the majority vote at the end: this is dominated by the other terms above

Dependence on n in sklearn?

kNN classifier
k = 1
MNIST data

Scikit-learn nearest-neighbor classifier

Linear dependence on n

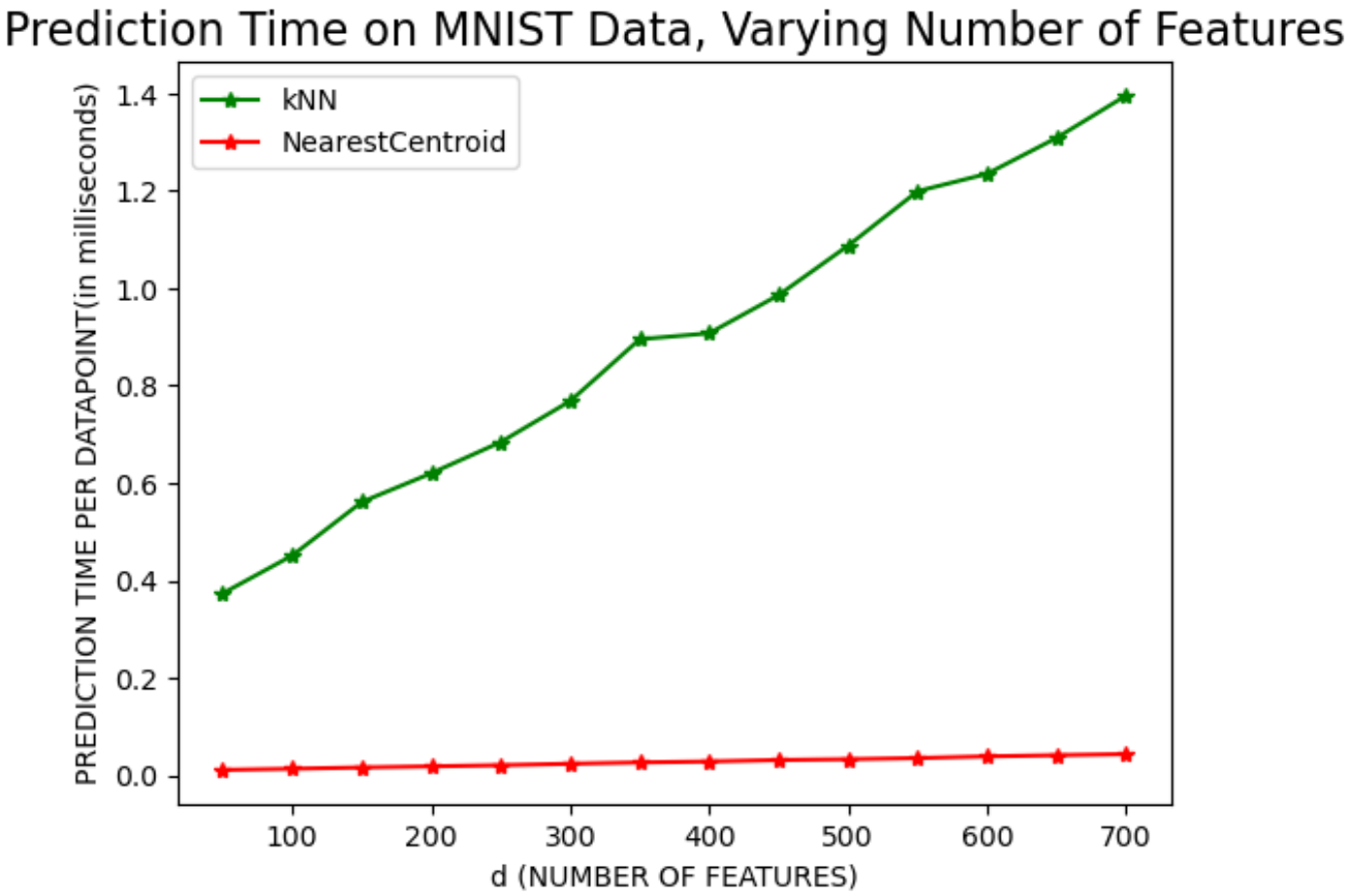


Dependence on d in sklearn?

kNN classifier
k = 1
ntrain = 5000
MNIST data

Scikit-learn
nearest-neighbor
and NC classifiers

Linear
dependence on d

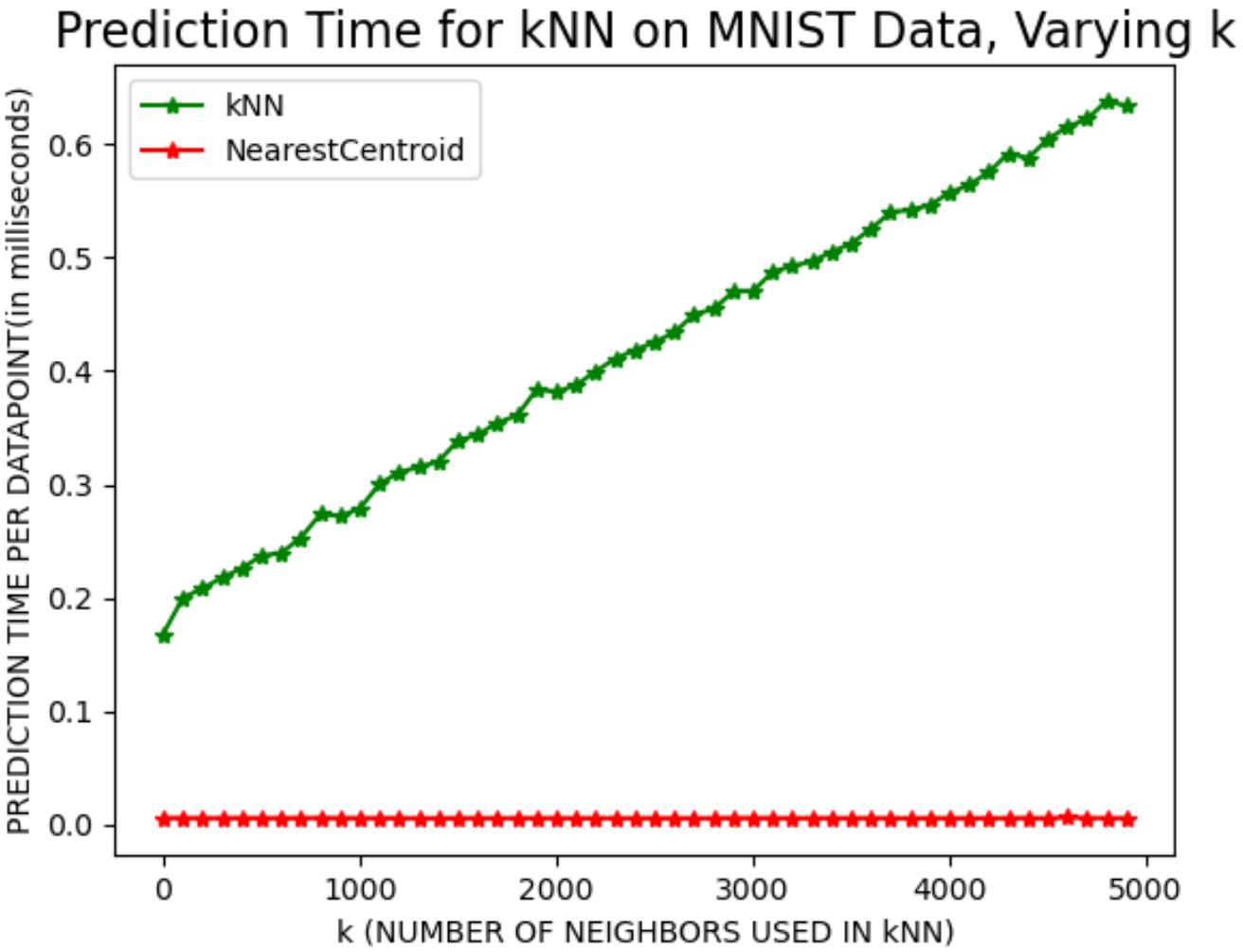


Dependence on k in sklearn?

kNN classifier
ntrain = 5000
MNIST data

Scikit-learn
nearest-neighbor
and NC classifiers

Linear
dependence on d



Questions?

k Nearest Neighbors

Time-Space Complexity of Classifiers

The Curse of Dimensionality

Comparing Classifiers

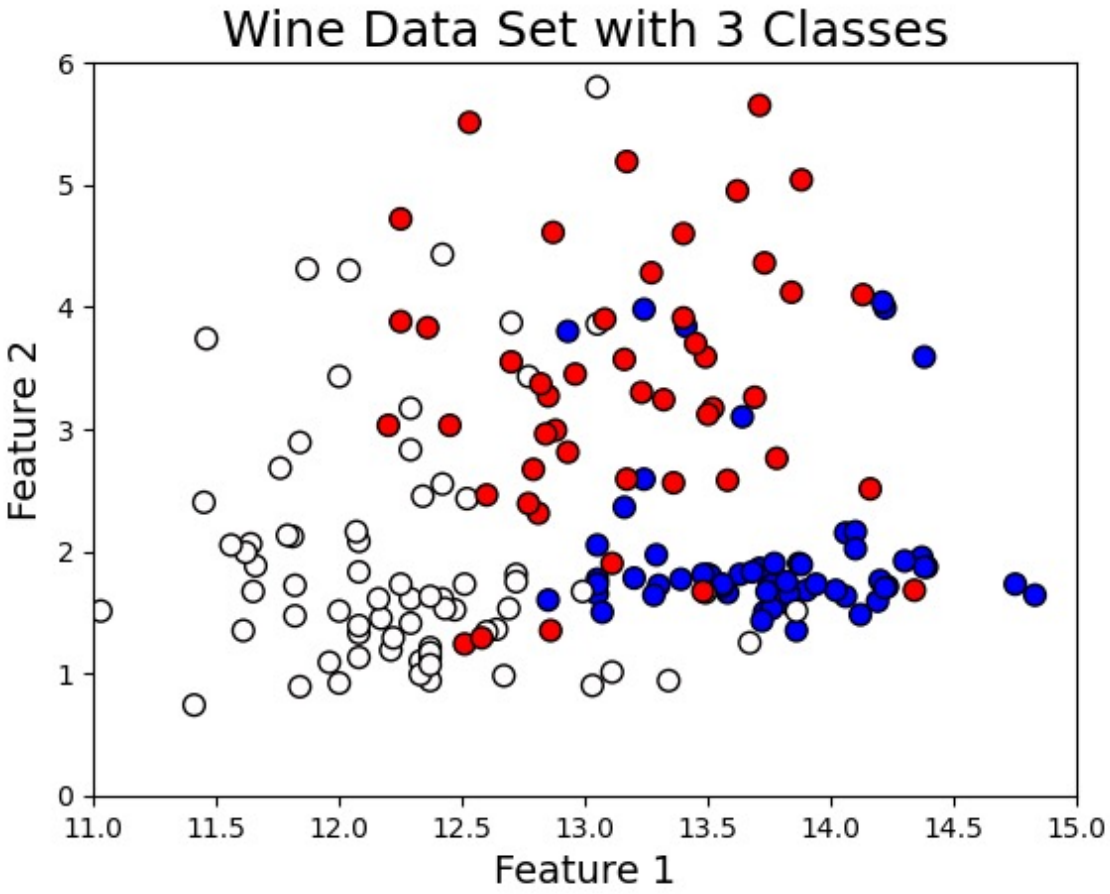
Curse of Dimensionality

- In machine learning the “curse of dimensionality” refers to the fact that classifiers that rely on distance or density may perform poorly in high dimensions

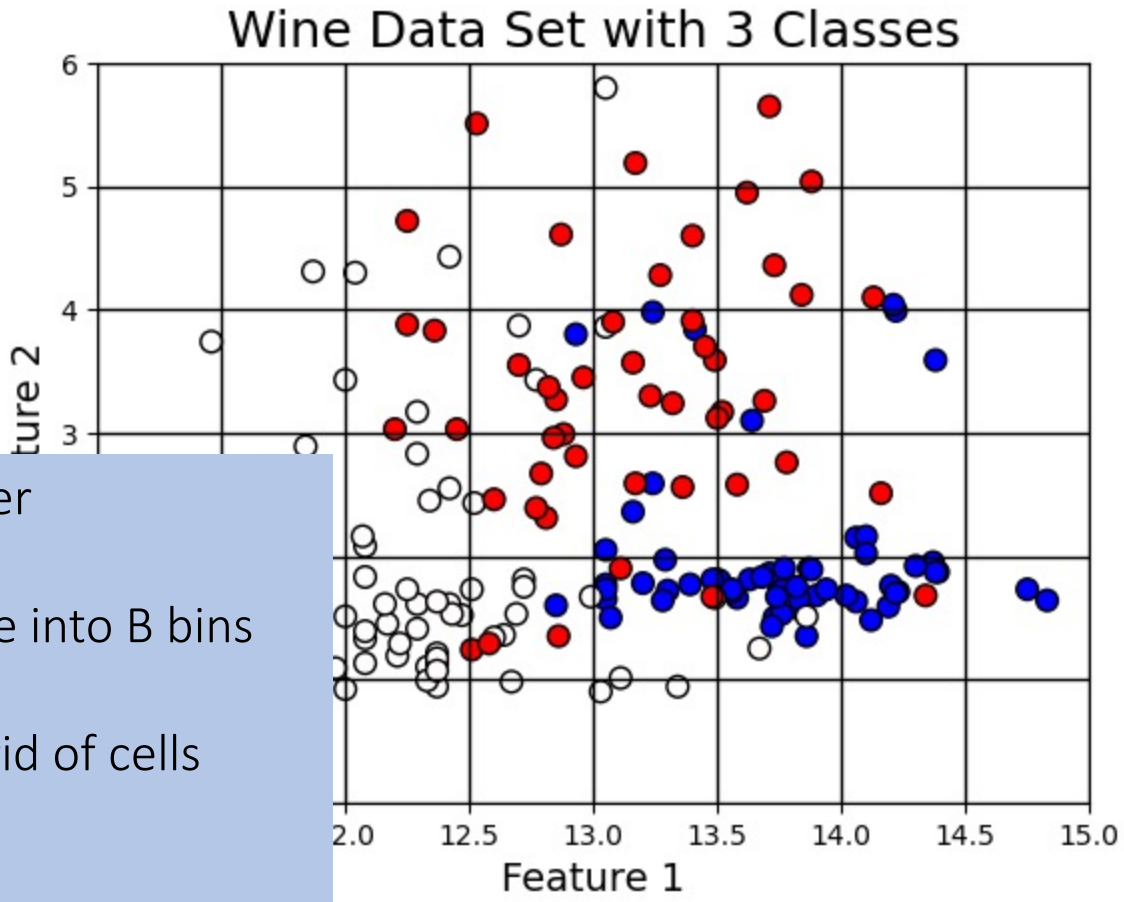
- We have already seen that distance-based methods can have increased error rates as dimensionality increases

- In the next few slides we look at another aspect of high-dimensional data, namely ”density” of data points

Wine Dataset



Wine Dataset with Bins



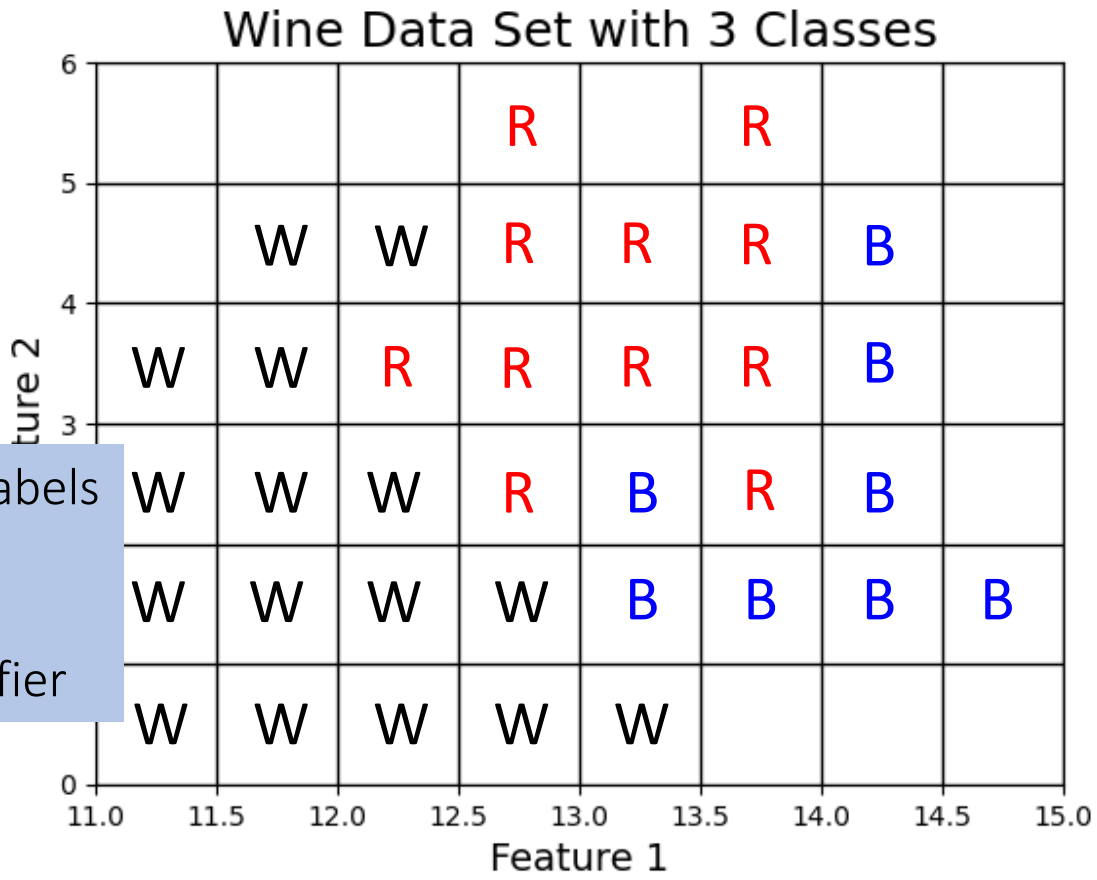
Binning Classifier

Bin each feature into B bins

Defines B x B grid of cells

Classifier?
Pick majority class in each cell

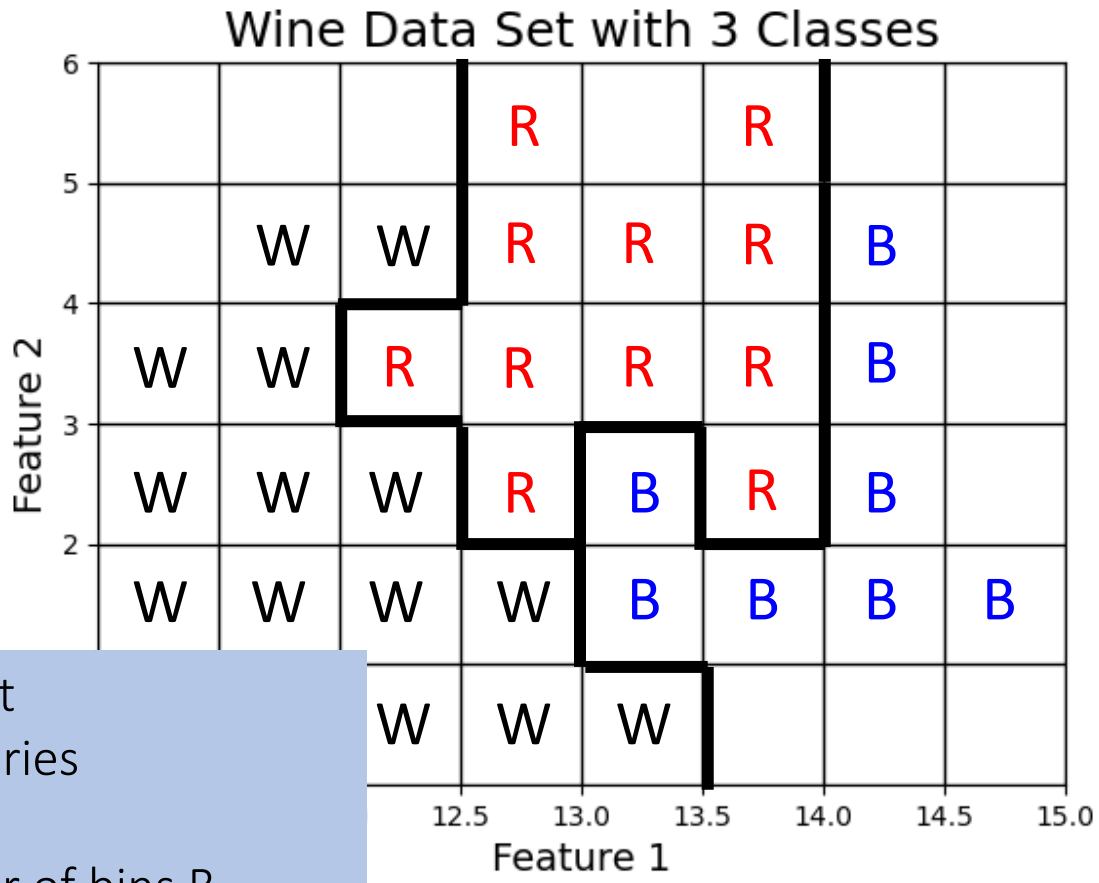
The Binning Classifier



Resulting class labels for bins

This is the classifier

The Binning Classifier



Resulting implicit decision boundaries

For large number of bins B boundaries can be very complex

The Binning Classifier

(This is more for illustration than a real classifier)

- Bin each feature into B bins, e.g., B=10
  - We can bin any way we like, e.g., equal-width bins over range
- In d-dimensions this gives us B x B x….. x B x B = B^d cells
- Training the Binning Classifier
  - On the training data, compute the majority class in each cell
  - Store this as a d-dimensional array
- Prediction with the Binning Classifier
  - Look-up what cell x belongs to, and predict the label for that cell

Problems with the Binning Classifier

- The main problem is that it does not scale with d (dimensionality)

- **The number of cells grows exponentially with d**, i.e., as B^d
  - e.g., say B = 10, then $B^2 = 100$, $B^5=10^5$, $B^{20} = 10^{20}$
  - Even with B=2, then B^{20} ~ 1 million, B^{40} ~ 1 billion, etc

- Problem 1: memory

- Problem 2: many many cells will have zero data points.

  Why? Say we have $B^d = 2^{40}$ = 1 billion cells, and n = 1 million datapoints
  -> only 1 million/1 billion = 1/million th of cells are non-zero (at best)
  -> so we will never have enough data to fill the cells as d grows

This is known as the "**curse of dimensionality**"
Distance-based methods are particularly sensitive to this issue

Extensions of Binning Classifiers

We could think of a few ways to improve the binning classifier, e.g.,

1. Put bins only where they are needed (differences between classes)
2. Only use features/dimensions where classes can be separated

Leads to the idea of tree-based and rule-based classifiers, where trees and rules can learned from data

We will see this later in the course when we discuss decision-tree classifiers

Questions?

k Nearest Neighbors

Time-Space Complexity of Classifiers

The Curse of Dimensionality

Comparing Classifiers

Baseline: Majority Classifier

- Always predict the most common label in the dataset
  - (assume label frequencies in train and test are roughly equal)

- What is the error rate of this "classifier"
  - Let maxP = probability (rel. frequency) of the most common label
  - Error of majority classifier = 1 – maxP
  - Why? …..

- This is not a real classifier, but its useful as a baseline
  - E.g., binary classification, P(Class 1) = 0.95
  - So the error rate of the majority classifier is 5%
  - Any classifier we build should have error rate at least as low as this

Example of a Majority Classifier

Training data with 100 examples:
60 from class 1, 30 from class 2, 10 from class 3

Class probabilities:
P(Class 1) = 0.6, P(Class 2) = 0.3, P(Class 3) = 0.1

Majority class is Class 1, with P(Class 1) = 0.6

Accuracy of Majority Classifier = P(Class 1) = 0.6
Error Rate of Majority Classifier = 1 – P(Class 1) = 0.4

Note: with C equally likely classes
Error Rate of Majority Classifier = 1 – 1/C = (C-1)/C
e.g., 90% for MNIST digits, and 66% for Wine

Classifiers discussed so far…

Majority Classifier: simple baseline, high error

Binned Classifier: simple baseline, impractical for d > 10 or 20

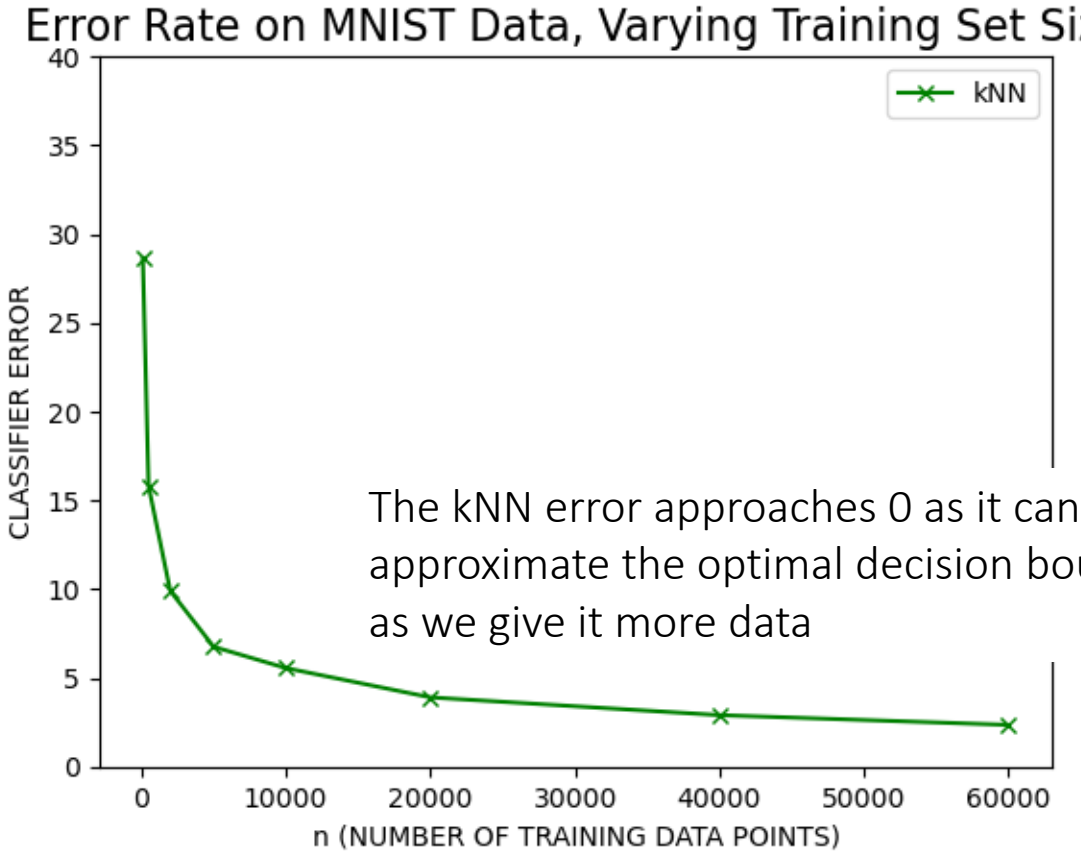
Nearest-Centroid: simple decision boundaries

kNN: can have complex decision boundaries (depends on k)

Comparing Classifiers

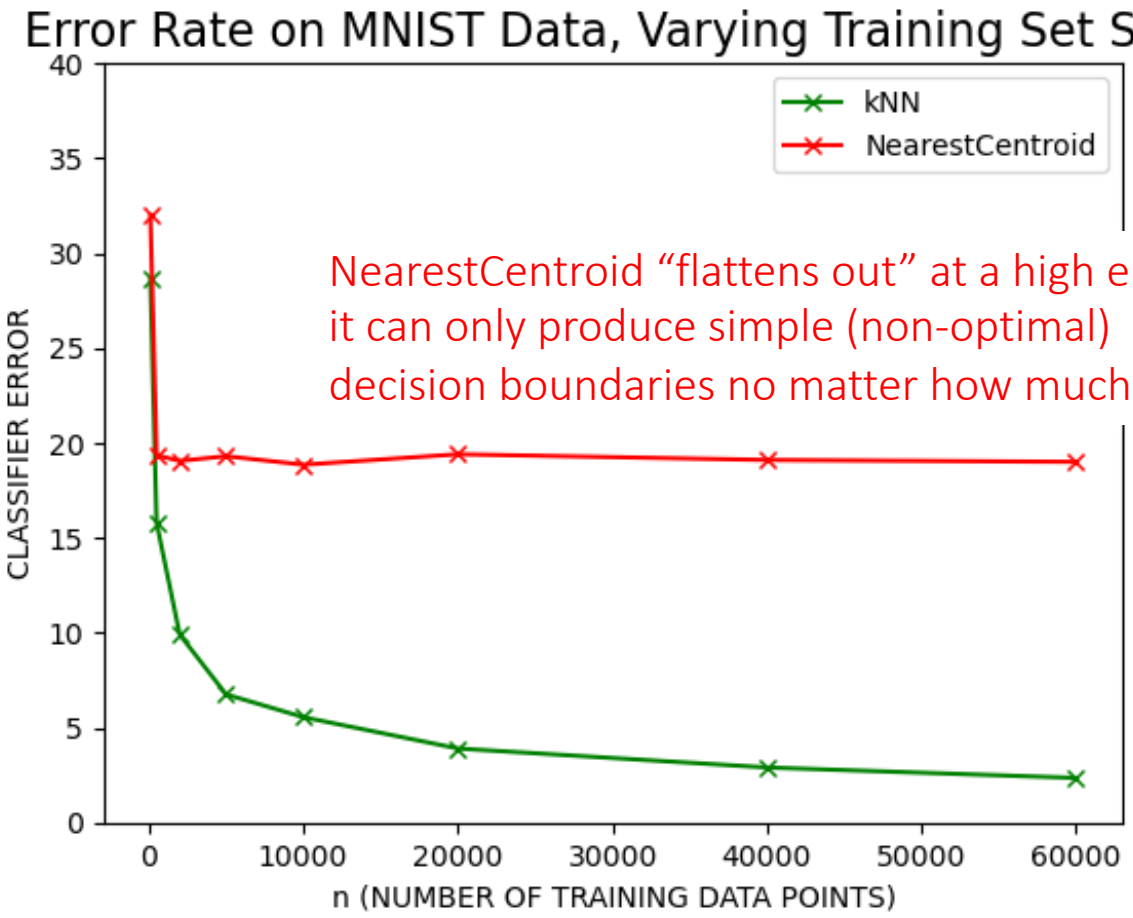
Classifier	Majority	Binned	Nearest Centroid	kNN
Prediction space complexity	O(1)	$O(B^d)$	O(C d)	O(nd)
Prediction time complexity	O(1)	O(1)	O(C d)	O(nd + nk)
Decision boundaries	None	Complex	Simple	Complex for small k
Distance-based?	No	No	Yes	Yes
Error on MNIST	90%	?	~ 20%	~ 3%
Error on Wine (2d)	66%	?	~ 22%	~ 22%

Learning Curve for kNN



The kNN error approaches 0 as it can better approximate the optimal decision boundaries as we give it more data

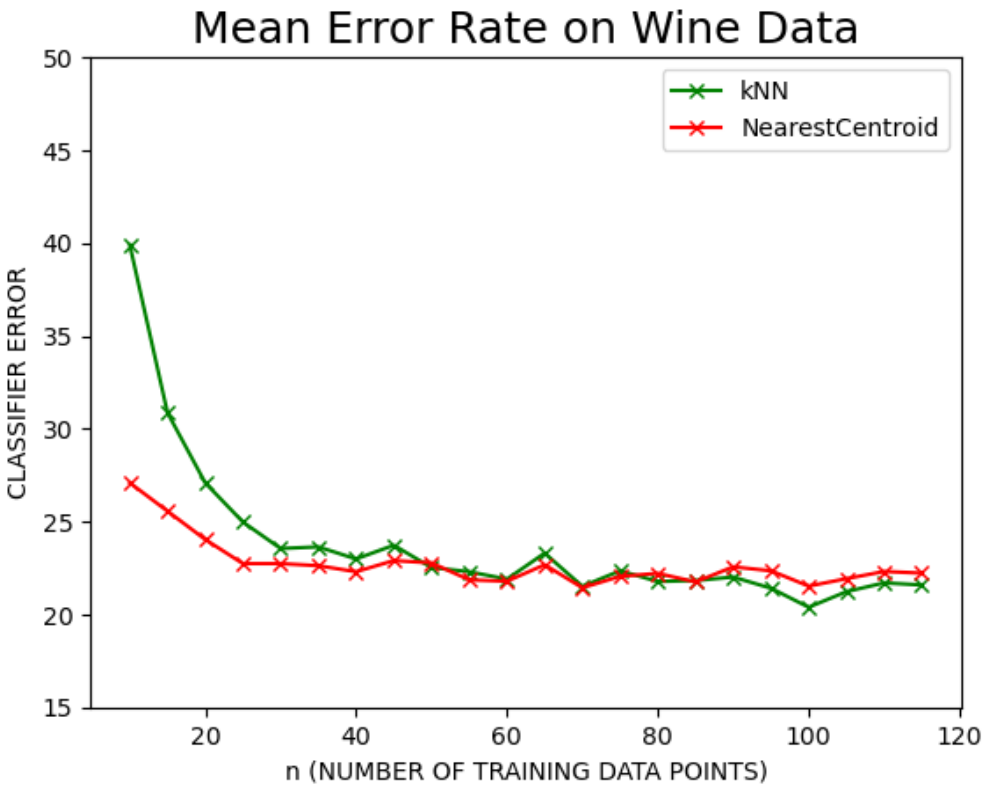
Learning Curve for kNN and NearestCentroid



NearestCentroid "flattens out" at a high error rate:
it can only produce simple (non-optimal)
decision boundaries no matter how much data we give it

On a different problem, the Wine Dataset, both kNN and NearestCentroid "flatten out" as training set increases

With more data we might see than kNN is less biased (closer to optimal) than NC, but we don't have enough data here to really tell.



Results generated as follows:
K = 5 for kNN
For each value of n
Dataset split into n training points and 40 test points
Repeated 100 times (randomly) and average reported

Summary and Wrapup

- k Nearest Neighbors
  - Number of neighbors k is a hyperparameter that can be tuned on a validation set
- Classifier time/space complexities
  - Important to consider how cost of using model scales with number of features (d), dataset size (n), and hyperparameters
- "Curse of Dimensionality"
  - Amount of "space" grows exponentially with number of features

Questions?
(Outside after lecture)