Nearest Neighbor Classification

Lecture 4

Machine Learning
Fall 2015

THE UNIVERSITY OF UTAH

Where are we? The roadmap

- Learning = search for functions
- Generalization is important
- Our first hypothesis
  - Decision trees
  - Learning algorithm: ID3

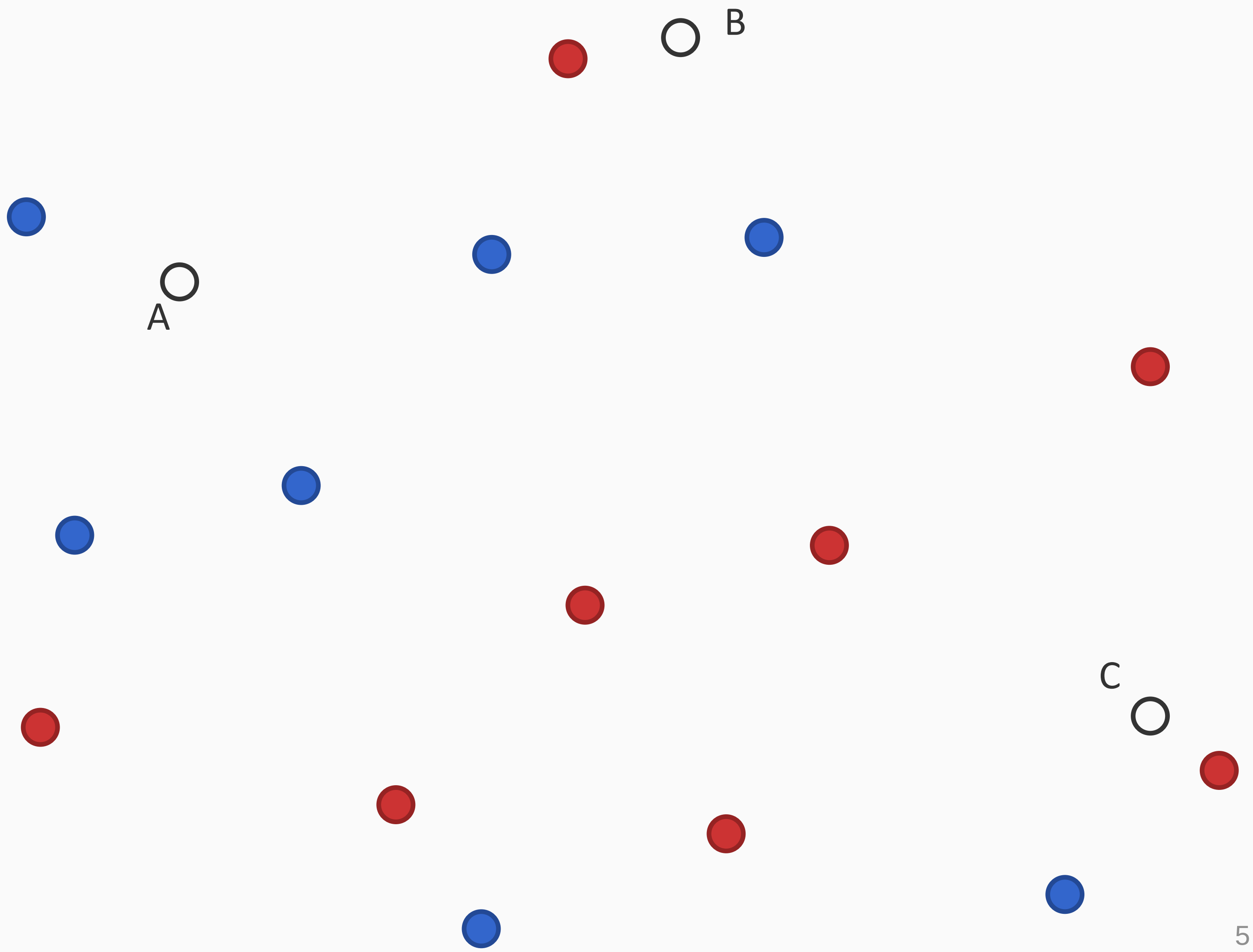
This lecture

- K-nearest neighbor classification
  - The basic algorithm
  - Different distance measures
  - Some practical aspects
- Voronoi Diagrams and Decision Boundaries
  - What is the hypothesis space?
- The Curse of Dimensionality

This lecture

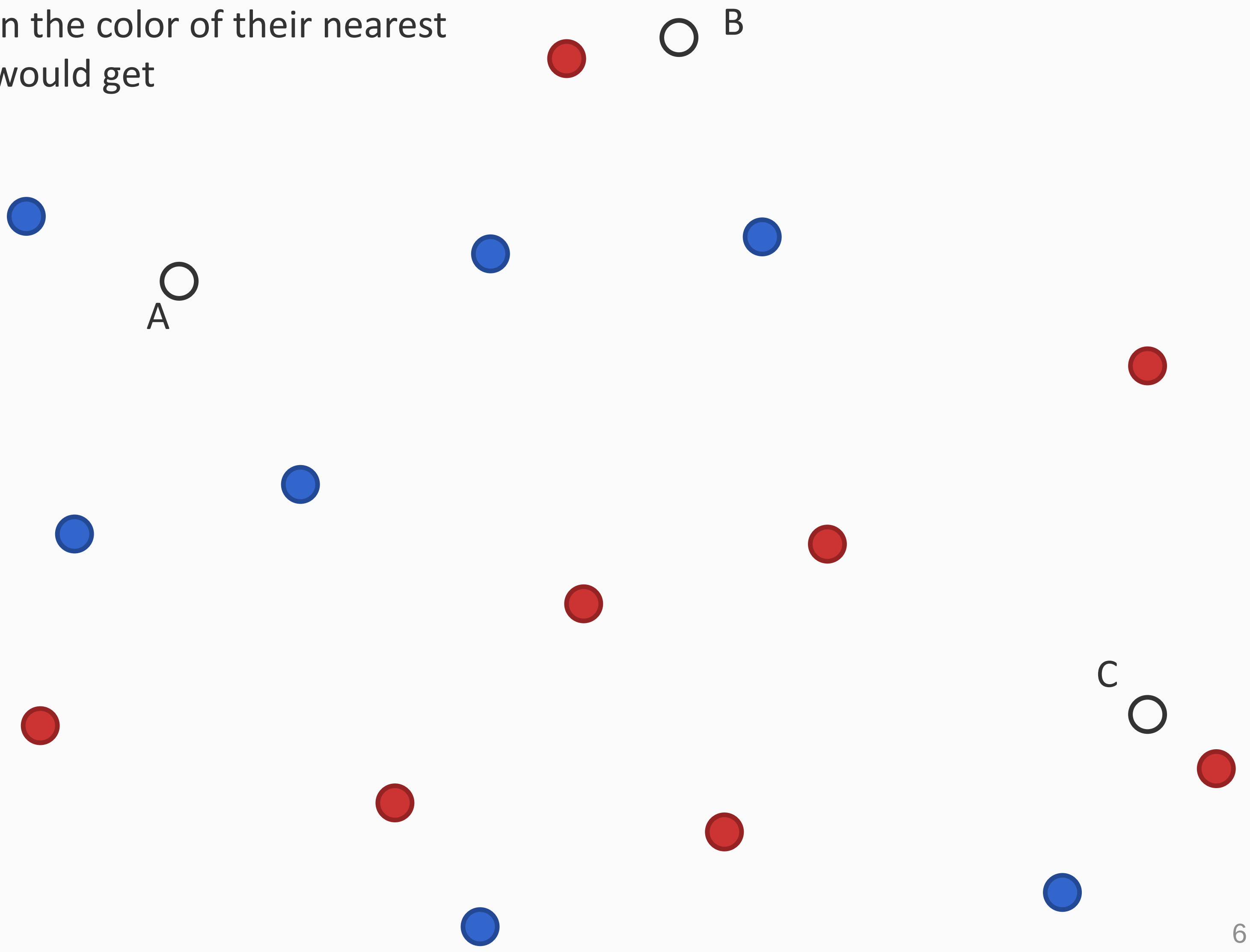
- K-nearest neighbor classification
  - The basic algorithm
  - Different distance measures
  - Some practical aspects
- Voronoi Diagrams and Decision Boundaries
  - What is the hypothesis space?
- The Curse of Dimensionality

How would you color the blank circles?

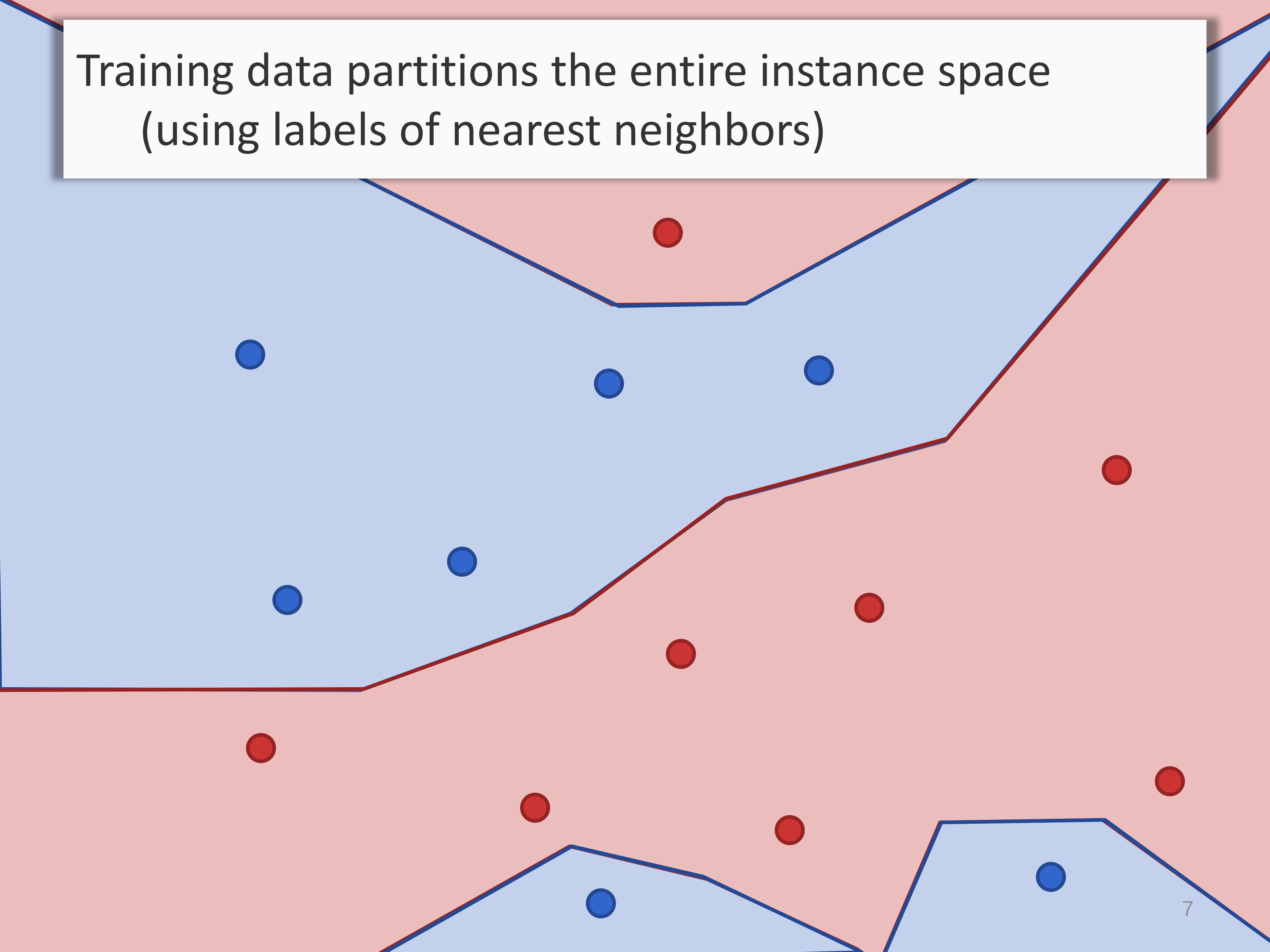


How would you color the blank circles?

If we based it on the color of their nearest neighbors, we would get
A: Blue
B: Red
C: Red



Training data partitions the entire instance space (using labels of nearest neighbors)



Nearest Neighbors: The basic version

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM
- Learning: Just store all the training examples
- Prediction for a new example $\mathbf{x}$
  - Find the training example $\mathbf{x}_i$ that is *closest* to $\mathbf{x}$
  - Predict the label of $\mathbf{x}$ to the label y_i associated with $\mathbf{x}_i$

K-Nearest Neighbors

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

- Prediction for a new example $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - Construct the label of $\mathbf{x}$ using these k points. *How?*
  - For classification: ?

K-Nearest Neighbors

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM
- Learning: Just store all the training examples
- Prediction for a new example $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - Construct the label of $\mathbf{x}$ using these k points. *How?*
  - For classification: Every neighbor votes on the label. Predict the most frequent label among the neighbors.
  - For regression: ?

K-Nearest Neighbors

- Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
  - E.g. $\mathbf{x}_i$ = a feature vector for an email, $\mathbf{y}_i$ = SPAM

- Learning: Just store all the training examples

- Prediction for a new example $\mathbf{x}$
  - Find the k *closest* training examples to $\mathbf{x}$
  - Construct the label of $\mathbf{x}$ using these k points. *How?*
  - For classification: Every neighbor votes on the label. Predict the most frequent label among the neighbors.
  - For regression: Predict the mean value

Instance based learning

- A class of learning methods
  - Learning: Storing examples with labels
  - Prediction: When presented a new example, classify the labels using *similar* stored examples

- K-nearest neighbors algorithm is an example of this class of methods

- Also called *lazy* learning, because most of the computation (in the simplest case, all computation) is performed only at prediction time

Questions?

Distance between instances

- In general, a good place to inject knowledge about the domain

- Behavior of this approach can depend on this

- How do we measure distances between instances?

Distance between instances

Numeric features, represented as n dimensional vectors

Distance between instances

Numeric features, represented as n dimensional vectors

- Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- L_p-norm
  - Euclidean = L_2
  - Manhattan = L_1
  - Exercise: What is L_∞?

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left(\sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p\right)^{\frac{1}{p}}$$

Distance between instances

What about symbolic/categorical features?

Distance between instances

Symbolic/categorical features

Most common distance is the *Hamming distance*

- Number of bits that are different
- Or: Number of features that have a different value
- Also called the *overlap*
- Example:

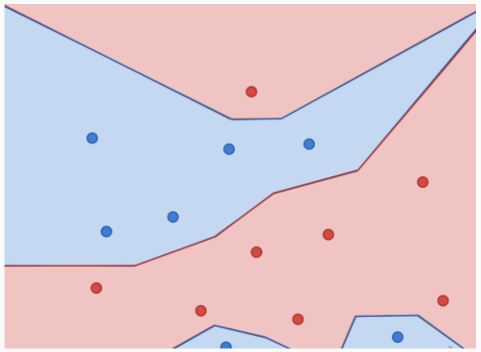
  $\mathbf{X}_1$: {Shape=Triangle, Color=Red, Location=Left, Orientation=Up}

  $\mathbf{X}_2$: {Shape=Triangle, Color=Blue, Location=Left, Orientation=Down}

  Hamming distance = 2

Advantages

- Training is *very fast*
  - Just adding labeled instances to a list
  - More complex indexing methods can be used, which slow down learning slightly to make prediction faster
- Can learn very complex functions



- We always have the training data
  - For decision trees, after training, we don't store the data anymore. What if we want to do something with it later…

Disadvantages

- Needs a lot of storage
  - Is this really a problem now?
- Prediction can be slow!
  - Naïvely: O(dN) for N training examples in d dimensions
  - More data will make it slower
  - Compare to decision trees, where prediction is very fast
- Nearest neighbors are fooled by irrelevant attributes
  - Important and subtle
  - Decision trees are less affected by this

Questions?

Summary: K-Nearest Neighbors

- Probably the first "machine learning" algorithm
  - Guarantee: If there are enough training examples, the error of the nearest neighbor classifier will converge to the error of the optimal (i.e. best possible) predictor
- In practice, use an odd K. Why?
  - To break ties
- How to choose K? Using a held-out set or by cross-validation
- Feature normalization could be important
  - Often, good idea to center the features to make them zero mean and unit standard deviation. Why?
  - Because different features could have different scales (weight, height, etc); but the distance weights them equally
- Variants exist
  - Neighbors' labels could be weighted by their distance

Detour: Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters.

How do we know what the best feature set and hyper-parameters are?

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

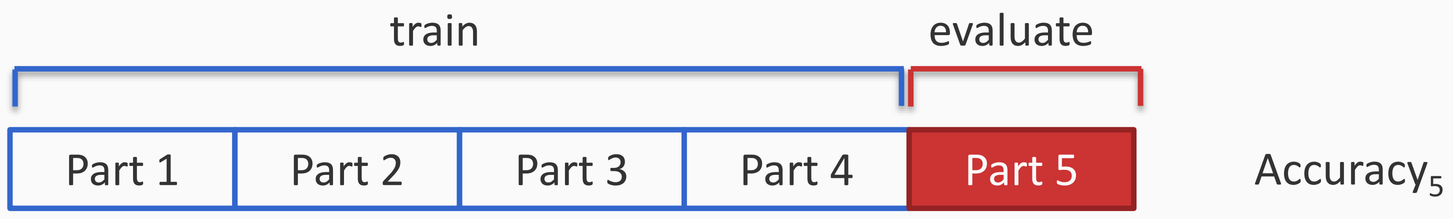
Part 1	Part 2	Part 3	Part 4	Part 5

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one



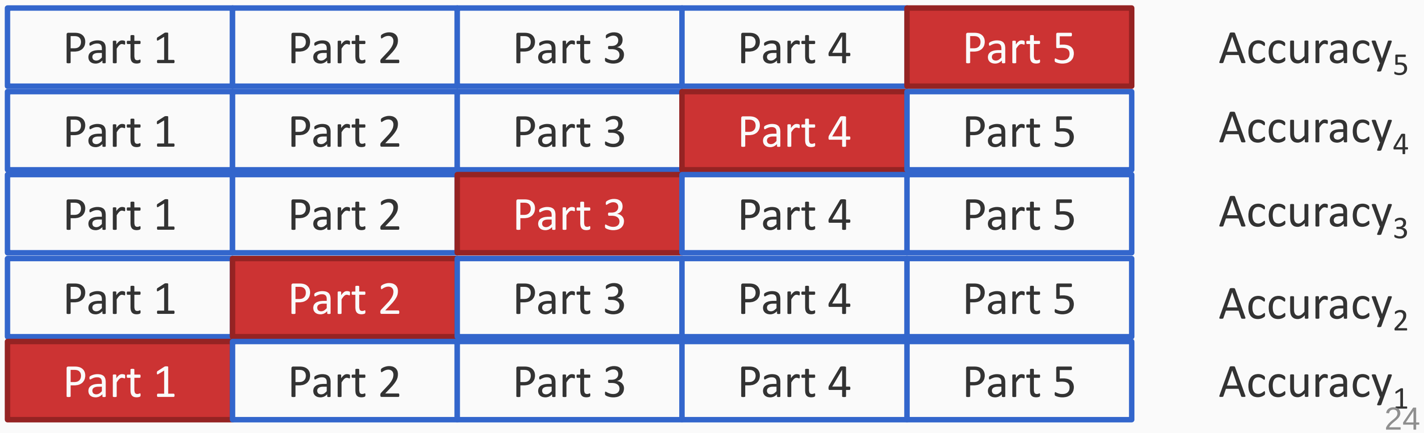
K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the *validation set*


Part 1 Part 2 Part 3 Part 4 Part 5 Accuracy_5
Part 1 Part 2 Part 3 Part 4 Part 5 Accuracy_4
Part 1 Part 2 Part 3 Part 4 Part 5 Accuracy_3
Part 1 Part 2 Part 3 Part 4 Part 5 Accuracy_2
Part 1 Part 2 Part 3 Part 4 Part 5 Accuracy_1

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the *validation set*

4. The quality of this feature set/hyper-parameter is the average of these K estimates

Performance = $(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5)/5$

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the *validation set*

4. The quality of this feature set/hyper-parameter is the average of these K estimates

Performance = (accuracy$_1$ + accuracy$_2$ + accuracy$_3$ + accuracy$_4$ + accuracy$_5$)/5

5. Repeat for every feature set/hyper parameter choice

Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters

How do we know what the best feature set and hyper-parameters are?

1. Evaluate every feature set and hyper-parameter using cross-validation (could be computationally expensive)

2. Pick the best according to cross-validation performance

3. Train on full data using this setting

Where are we?

- K-nearest neighbor classification
  - The basic algorithm
  - Different distance measures
  - Some practical aspects
- Voronoi Diagrams and Decision Boundaries
  - What is the hypothesis space?
- The Curse of Dimensionality

Where are we?

- K-nearest neighbor classification
  - The basic algorithm
  - Different distance measures
  - Some practical aspects
- Voronoi Diagrams and Decision Boundaries
  - What is the hypothesis space?
- The Curse of Dimensionality

Decision Boundaries

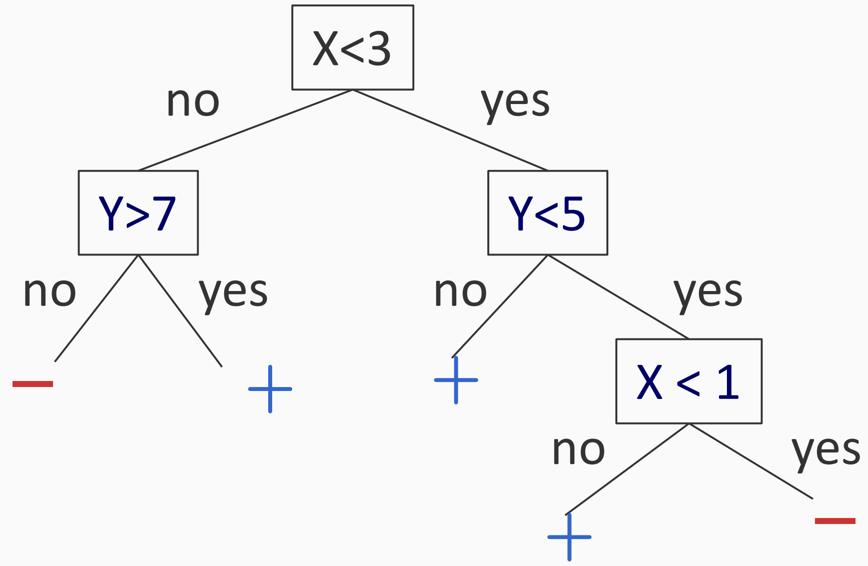
- Recall: Learning = Finding a good approximation of an unknown oracle function
  - Requires search over functions

Is this the same as *"Learning = generalization?"*

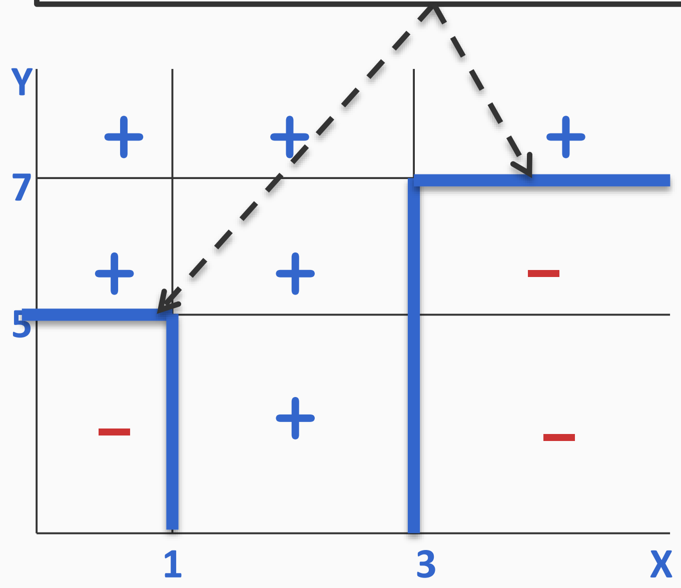
- One way to visualize learned models for binary classification: *"Which elements of the instance space will it classify as positive or negative?"*

- *Decision boundary*: The line (surface, in general) separating the positive and negative regions
  - Helps to visualize the complexity of learned functions models

We have already seen decision boundaries in the context of decision trees



Decision boundaries can be non-linear



If numeric features are thresholded, the decision boundary will be comprised of axis parallel lines

The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

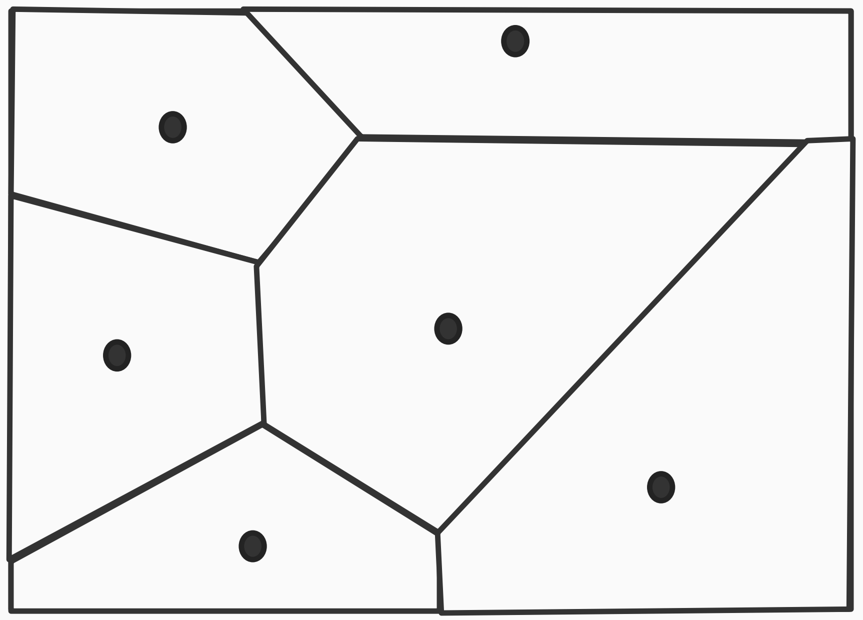
The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

- **No,** it never forms an explicit hypothesis

But we can still ask: Given a training set what is the implicit function that is being computed

The Voronoi Diagram

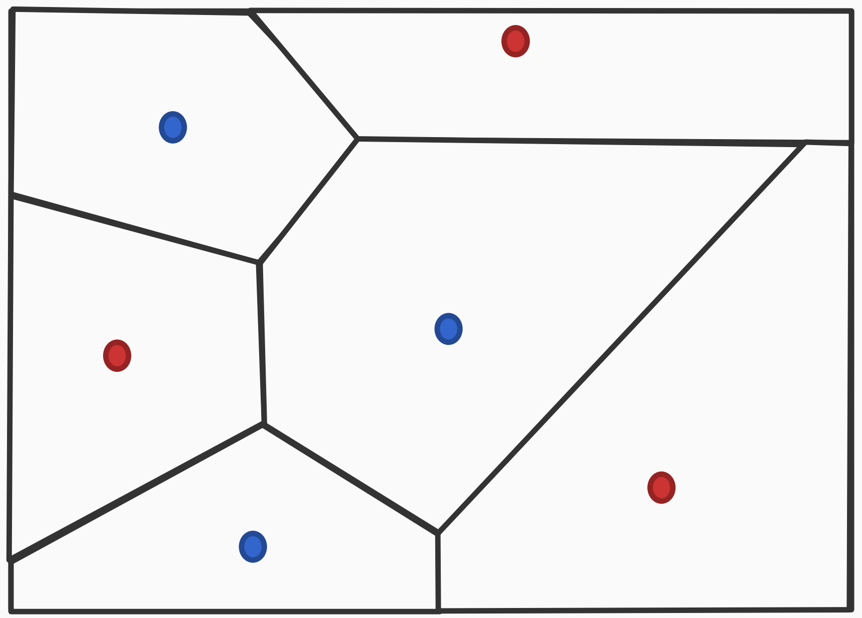


For any point **x** in a training set S, the Voronoi Cell of **x** is a polyhedron consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

- Covers the entire space

Voronoi diagrams of training examples



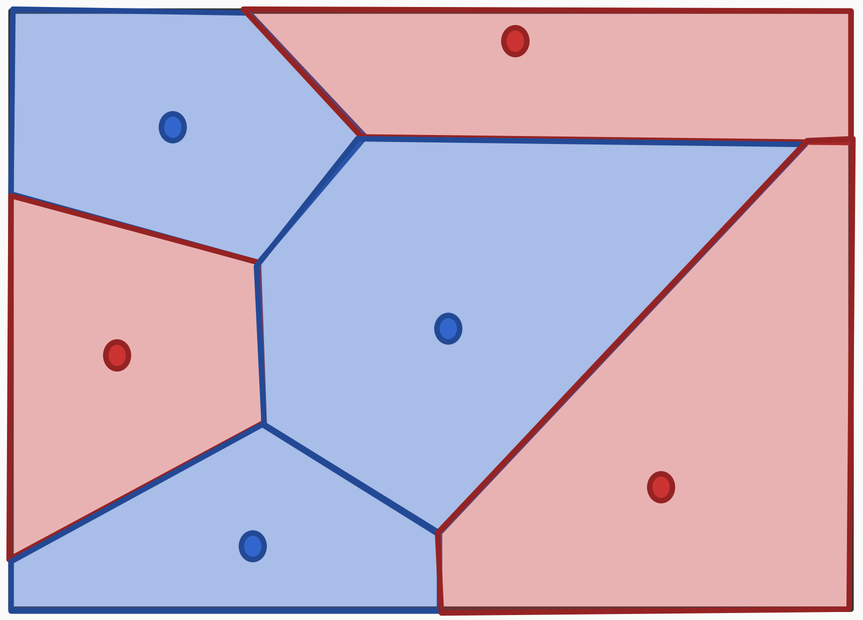
Points in the Voronoi cell of a training example are closer to it than any others

For any point **x** in a training set S, the Voronoi Cell of **x** is a polytope consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

- Covers the entire space

Voronoi diagrams of training examples



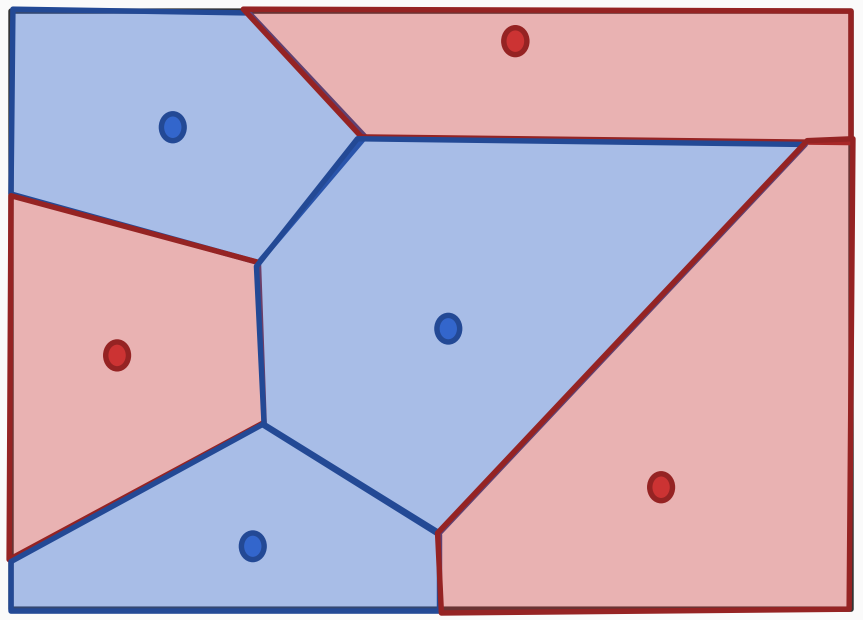
Points in the Voronoi cell of a training example are closer to it than any others

For any point **x** in a training set S, the Voronoi Cell of **x** is a polyhedron consisting of all points closer to **x** than any other points in S

The Voronoi diagram is the union of all Voronoi cells

- Covers the entire space

Voronoi diagrams of training examples



Points in the Voronoi cell of a training example are closer to it than any others

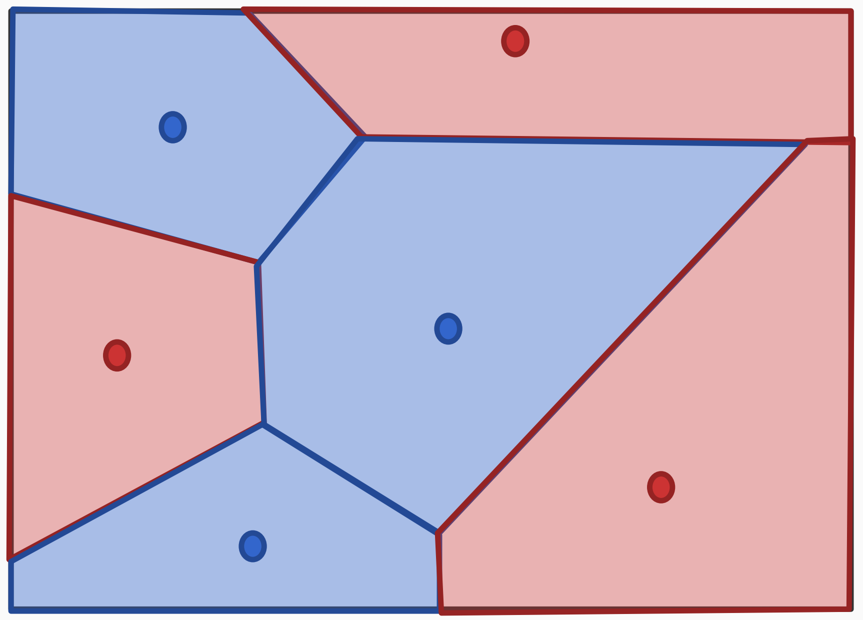
Picture uses Euclidean distance with 1-nearest neighbors.

What about K-nearest neighbors?

Also partitions the space, but much more complex decision boundary

Voronoi diagrams of training examples

What about points on the boundary? What label will they get?



Points in the Voronoi cell of a training example are closer to it than any others

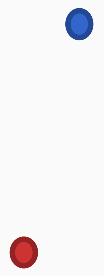
Picture uses Euclidean distance with 1-nearest neighbors.

What about K-nearest neighbors?

Also partitions the space, but much more complex decision boundary

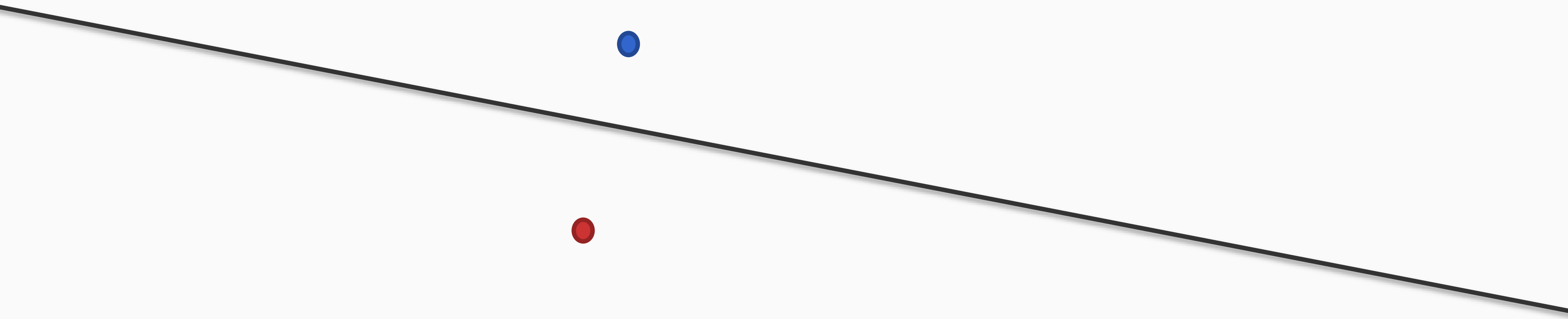
Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



– A line bisecting the two points

This lecture

- K-nearest neighbor classification
  - The basic algorithm
  - Different distance measures
  - Some practical aspects
- Voronoi Diagrams and Decision Boundaries
  - What is the hypothesis space?
- The Curse of Dimensionality

Why your classifier might go wrong

Two important considerations with learning algorithms

- Overfitting: We have already seen a version of this
- The curse of dimensionality
  - Methods that work with low dimensional spaces may fail in high dimensions
  - What might be intuitive for 2 or 3 dimensions do not always apply to high dimensional spaces

Check out the 1884 book *Flatland: A Romance of Many Dimensions* for a fun introduction to the fourth dimension

Of course, irrelevant attributes will hurt

Suppose we have 1000 dimensional feature vectors

- But only 10 features are relevant

- Distances will be dominated by the large number of irrelevant features

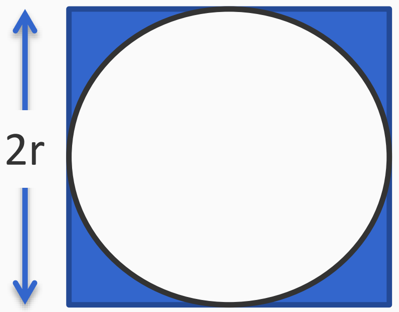
But even with only relevant attributes, high dimensional spaces behave in odd ways

The Curse of Dimensionality

Intuitions that are based on 2 or 3 dimensional spaces do not always carry over to high dimensional spaces

Example 1: What fraction of the points in a cube lie outside the sphere inscribed in it?

In two dimensions



What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

$$1 - \frac{\pi r^2}{4r^2} = 1 - \frac{\pi}{4}$$

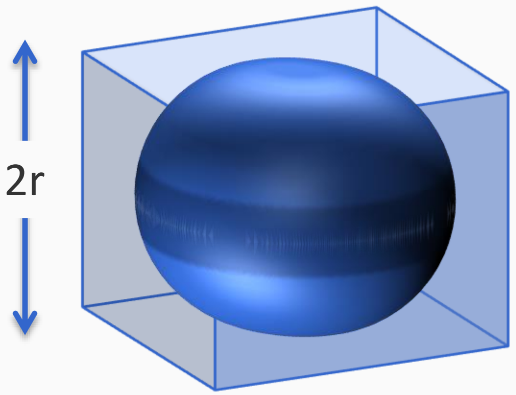
But, distances do not behave the same way in high dimensions

The Curse of Dimensionality

Intuitions that are based on 2 or 3 dimensional spaces do not always carry over to high dimensional spaces

Example 1: What fraction of the points in a cube lie outside the sphere inscribed in it?

In three dimensions



What fraction of the cube is outside the inscribed sphere in three dimensions?

$$1 - \frac{\frac{4}{3}\pi r^3}{8r^3} = 1 - \frac{\pi}{6}$$

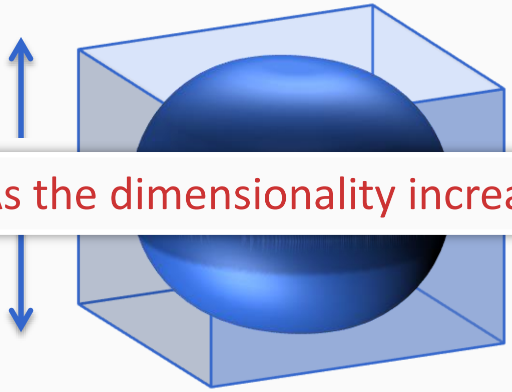
But, distances do not behave the same way in high dimensions

The Curse of Dimensionality

Intuitions that are based on 2 or 3 dimensional spaces do not always carry over to high dimensional spaces

Example 1: What fraction of the points in a cube lie outside the sphere inscribed in it?

In three dimensions



What fraction of the cube is outside the inscribed sphere in three dimensions?

$$\frac{4\ \pi r^3}{8r^3} \quad \frac{\pi}{6}$$

As the dimensionality increases, this fraction approaches 1!!

In high dimensions, most of the volume of the cube is far away from the center!

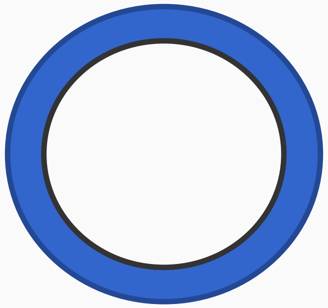
Questions?

The Curse of Dimensionality

Intuitions that are based on 2 or 3 dimensional spaces do not always carry over to high dimensional spaces

Example 2: What fraction of the volume of a unit sphere lies between radius 1 - ϵ and radius 1?

In two dimensions



What fraction of the area of the circle is in the blue region?

$$\frac{\pi \cdot 1^2 - \pi(1-\epsilon)^2}{\pi \cdot 1^2} = 1 - (1-\epsilon)^2$$

But, distances do not behave the same way in high dimensions

The Curse of Dimensionality

Intuitions that are based on 2 or 3 dimensional spaces do not always carry over to high dimensional spaces

Example 2: What fraction of the volume of a unit sphere lies between radius 1 - ϵ and radius 1?

But, distances do not behave the same way in high dimensions

In d dimensions, the fraction is $1 - (1 - \epsilon)^d$

As d increases, this fraction goes to 1!

In high dimensions, most of the volume of the sphere is far away from the center!

Questions?

The Curse of Dimensionality

- Most of the points in high dimensional spaces are far away from the origin!
  - In 2 or 3 dimensions, most points are near the center
  - Need more data to “fill up the space”

- Bad news for nearest neighbor classification in high dimensional spaces

Even if most/all features are relevant, in high dimensional spaces, most points are equally far from each other!

“Neighborhood” becomes very large

Presents computational problems too

Dealing with the curse of dimensionality

- Most “*real-world*” data is not uniformly distributed in the high dimensional space
  - Different ways of capturing the *underlying dimensionality* of the space
  - We will see dimensionality reduction later in the semester

- Feature selection, an art
  - Different methods exist
  - Select features, maybe by information gain
  - Try out different feature sets of different sizes and pick a good set based on a validation set

- Prior knowledge or preferences about the hypotheses can also help

Questions?

Summary: Nearest neighbors classification

- Probably the oldest and simplest learning algorithm
  - Prediction is expensive.
    - Efficient data structures help. k-D trees: the most popular, works well in low dimensions
    - Approximate nearest neighbors may be good enough some times. Hashing based algorithms exist
- Requires a distance measure between instances
  - Metric learning: Learn the “right” distance for your problem
- Partitions the space into a Voronoi Diagram
- Beware the curse of dimensionality

Questions?

Exercises

1. What will happen when you choose K to the number of training examples?

2. Suppose you want to build a nearest neighbors classifier to predict whether a beverage is a coffee or a tea using two features: the volume of the liquid (in milliliters) and the caffeine content (in grams). You collect the following data:

Volume (ml)	Caffeine (g)	Label
238	0.026	Tea
100	0.011	Tea
120	0.040	Coffee
237	0.095	Coffee

What is the label for a test point with Volume = 120, Caffeine = 0.013?

Why might this be incorrect?

How would you fix the problem?

Exercises

1. What will happen when you choose K to the number of training examples? *The label will always be the most common label in the training data*

2. Suppose you want to build a nearest neighbors classifier to predict whether a beverage is a coffee or a tea using two features: the volume of the liquid (in milliliters) and the caffeine content (in grams). You collect the following data:

Volume (ml)	**Caffeine (g)**	**Label**
238	0.026	Tea
100	0.011	Tea
120	0.040	Coffee
237	0.095	Coffee

What is the label for a test point with Volume = 120, Caffeine = 0.013?
Coffee

Why might this be incorrect?

Because Volume will dominate the distance

How would you fix the problem?

Rescale the features. Maybe to zero mean, unit variance