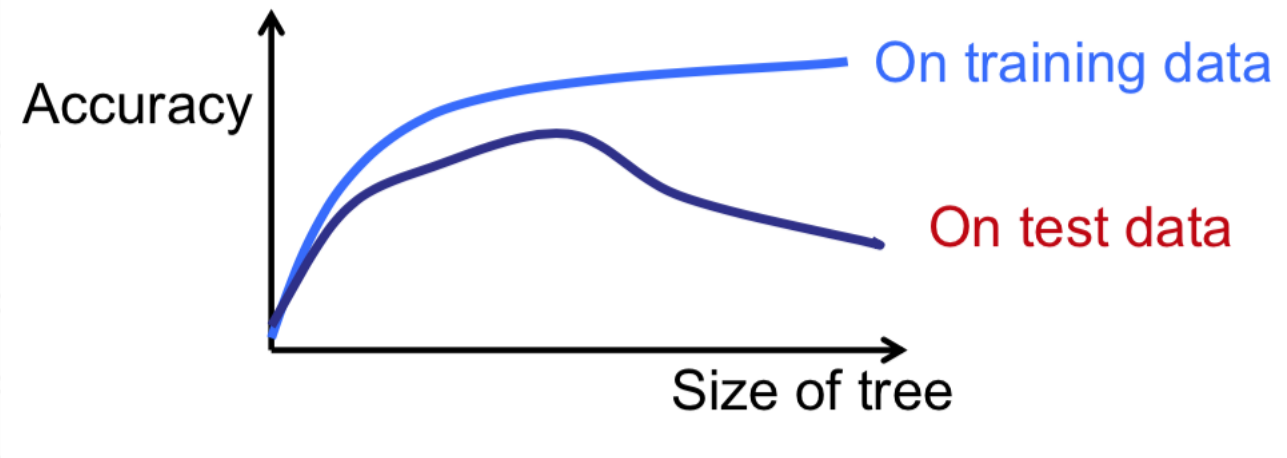
CMSC 422 Introduction to Machine Learning

Lecture 6 Geometry and Nearest Neighbors

Furong Huang / furongh@umd.edu

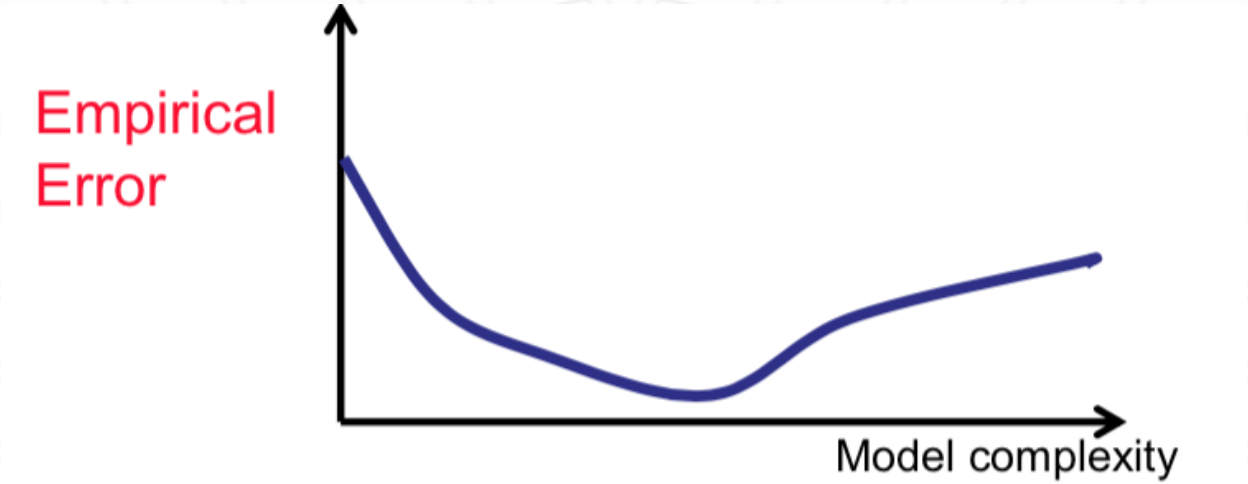
UNIVERSITY OF MARYLAND

Overfitting



- A decision tree overfits the training data when its accuracy on the training data goes up but its accuracy on unseen data goes down.

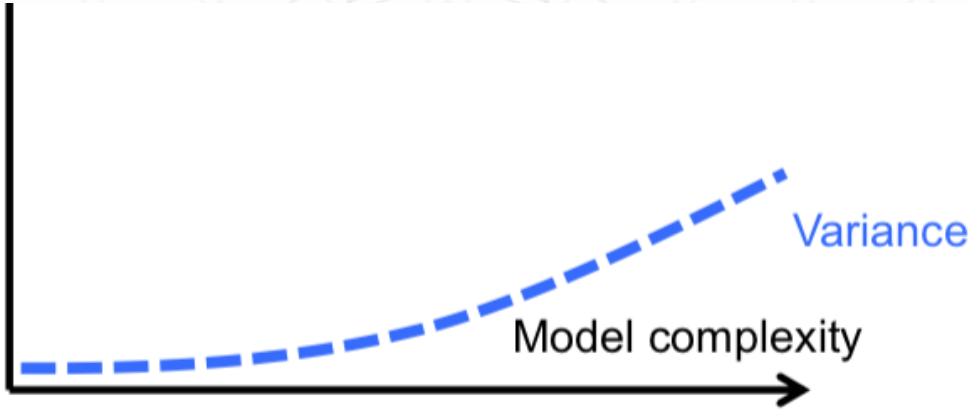
Overfitting



- Empirical error (= on a given data set):

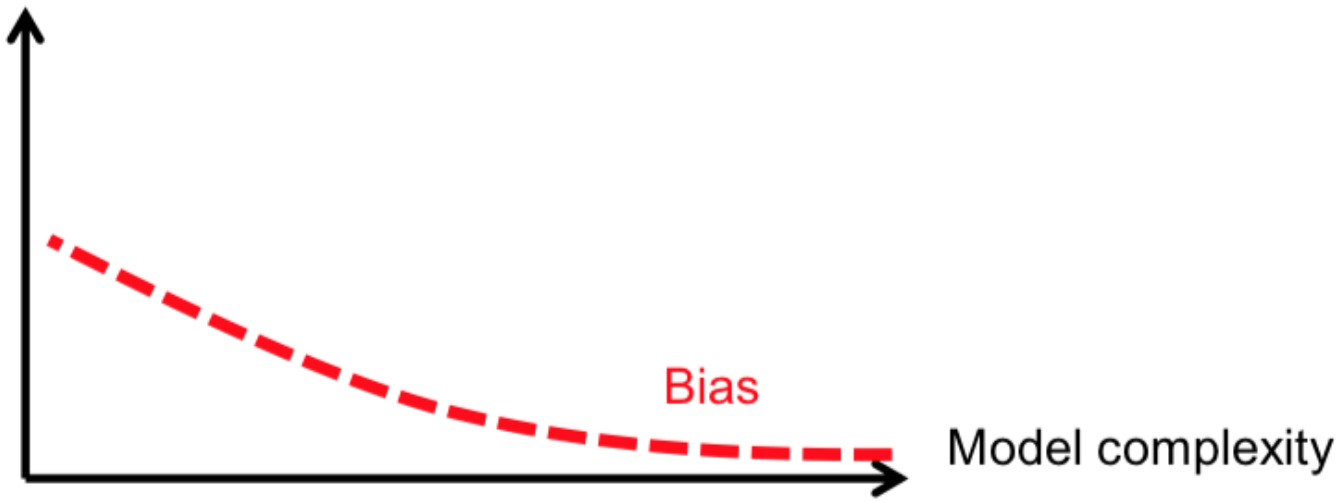
The percentage of items in this data set are misclassified by the classifier *f.*

Variance of A Learner (informally)



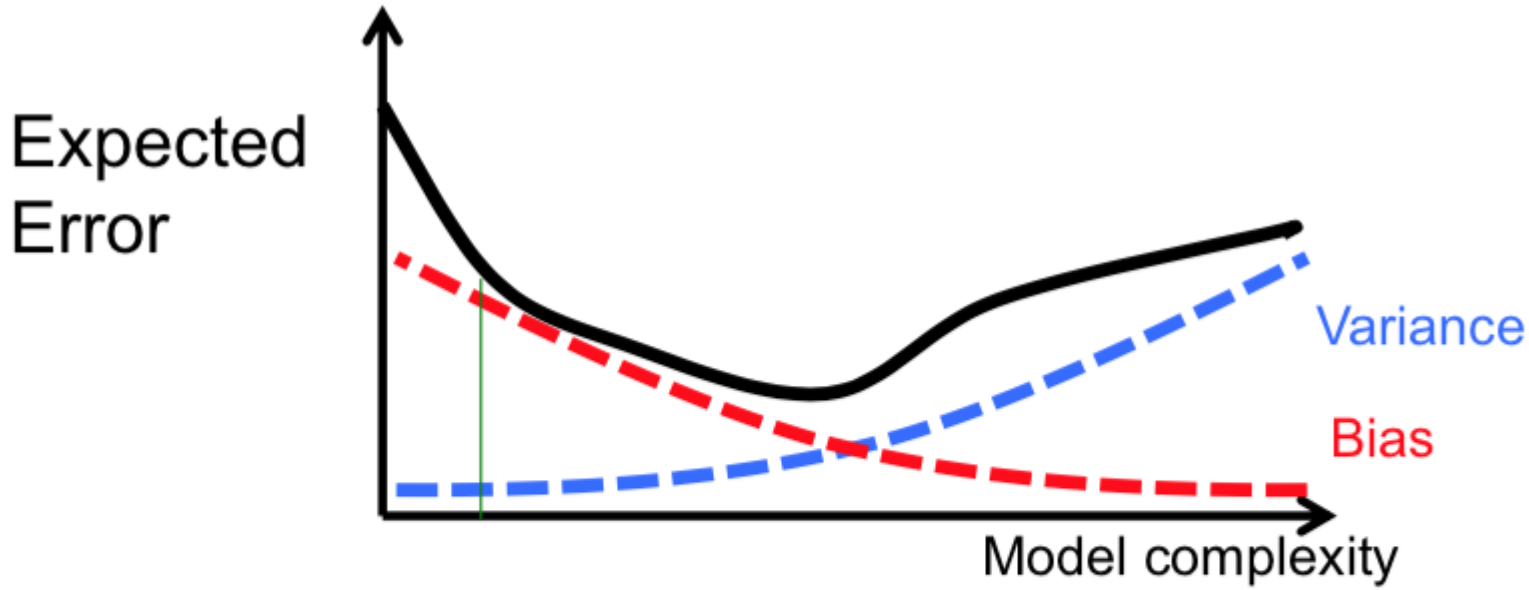
- How susceptible is the learner to minor changes in the training data?
  - (i.e. to different samples from P(X, Y))
- Variance increases with model complexity

Bias of A Learner (informally)



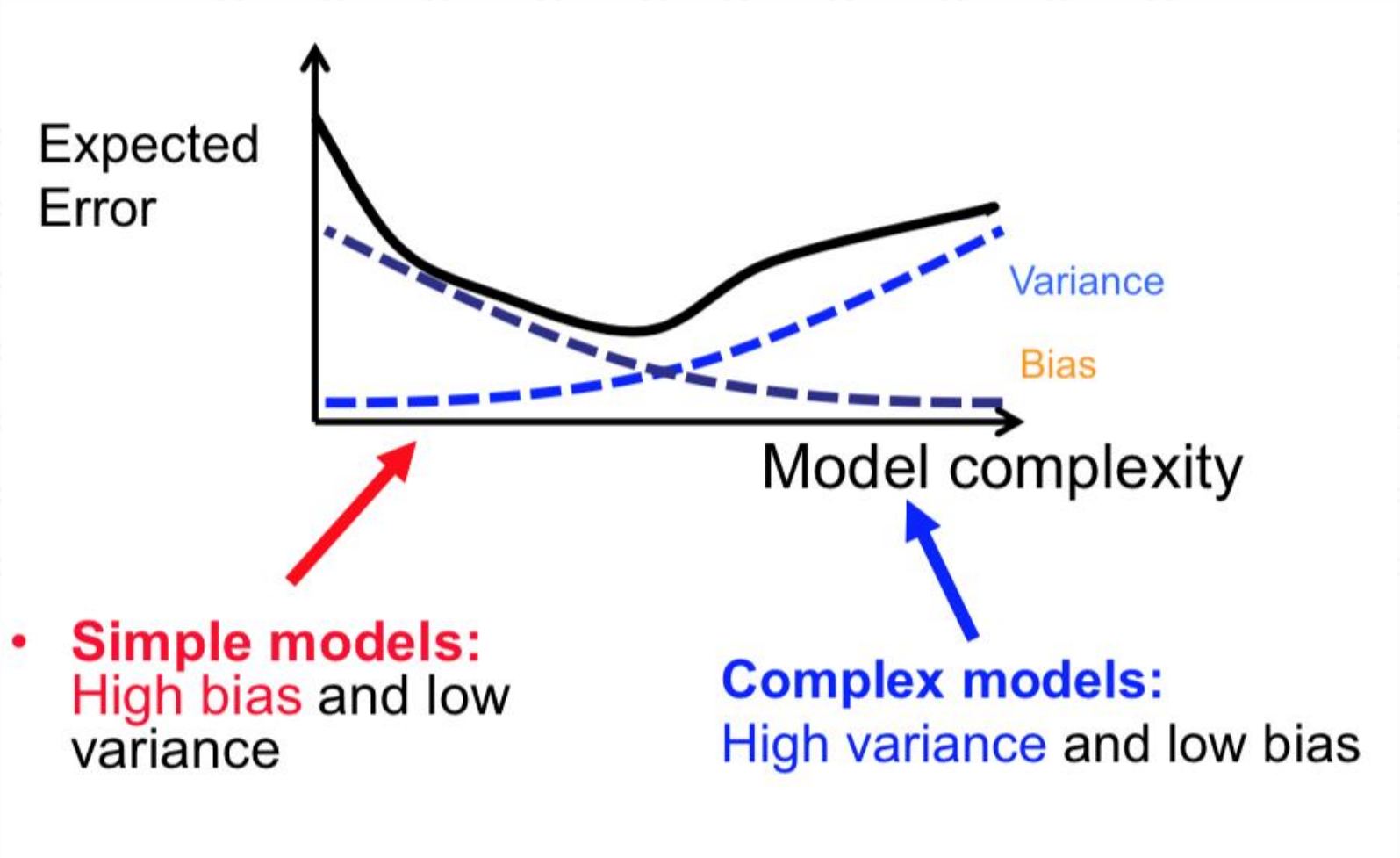
- How likely is the learner to identify the target hypothesis?
- Bias is low when the model is expressive (low empirical error)
- Bias is high when the model is (too) simple
  - The larger the hypothesis space is, the easier it is to be close to the true hypothesis.

Impact of Bias and Variance

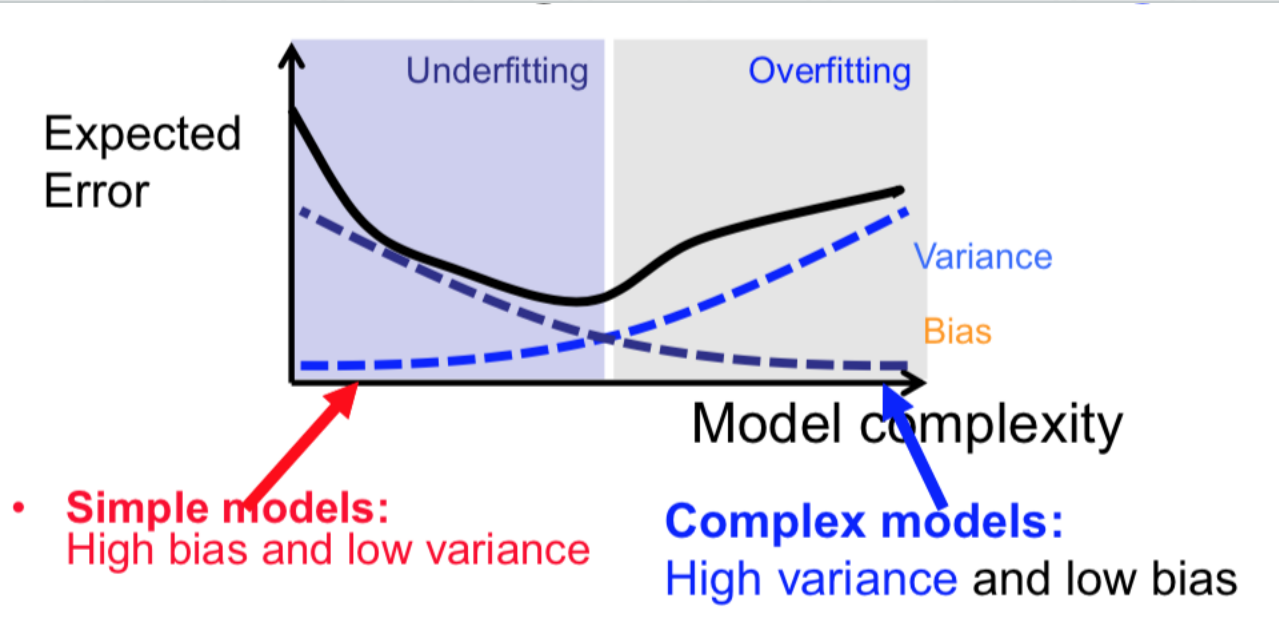


- Expected error ≈ bias + variance

Model Complexity



Underfitting and Overfitting



- This can be made more accurate for some loss functions.
- We will develop a more precise and general theory that trades expressivity of models with empirical error

Expectation

- X is a discrete random variable with distribution $P(X)$:
- Expectation of X ($\mathbb{E}[X]$), aka. the mean of X (μ_X)

$$\mathbb{E}[X] = \sum_x P(X = x)x := \mu_X$$

- Expectation of a function of X ($\mathbb{E}[f(X)]$)

$$\mathbb{E}[f(X)] = \sum_x P(X = x)f(X = x)$$

- If X is continuous, replace sums with integrals

Variance and Standard Deviation

- Squared difference between X and its mean:

$$(X - \mathbb{E}[X])^2 = (x - \mu_x)^2$$

- Variance of X:

$$\mathrm{V}ar[X] = \mathbb{E}[(X - \mu_x)^2] = \sigma_X^2$$

- The expected value of the square difference between X and its mean
- Standard deviation of X:

$$\sigma_X = \sqrt{\sigma_X^2} = \sqrt{Var(X)}$$

- (= the square root of the variance)

Variance (continued)

- The variance of X is equal to the expected value of X^2 minus the square of its mean

$$Var[X] = \mathbb{E}[X] - (\mathbb{E}[X])^2$$
$$= \mathbb{E}[X^2] - \mu_X^2$$

- Proof:

$$Var[X] = \mathbb{E}[(X - \mu_X)^2]$$
$$= \mathbb{E}[X^2 - 2\mu_X X + \mu_X^2]$$
$$= \mathbb{E}[X^2] - 2\mu_X \mathbb{E}[X] + \mu_X^2$$
$$= \mathbb{E}[X^2] - \mu_X^2$$

Practical impact on decision tree learning

What we want:

- A decision tree that neither underfits nor overfits
- Because it is expected to do best in the future

How can we encourage that behavior?

- Set a maximum tree depth D

Your thoughts?

What are the pros and cons of decision trees?

DEALING WITH DATA

Train/Dev/Test Sets

In practice, we always split examples into 3 distinct sets

- **Training set**
  - Used to learn the **parameters** of the ML model
  - e.g., what are the nodes and branches of the decision tree
- **Development set**
  - aka tuning set, aka validation set, aka held-out data
  - Used to learn **hyperparameters**
    - Parameter that controls other parameters of the model
    - e.g., max depth of decision tree
- **Test set**
  - Used to evaluate how well we're doing on new unseen examples

Cardinal rule of machine learning:

Never *ever* touch your test data!

Summary: what you should know

- **Decision Trees**
  - What is a decision tree, and how to induce it from data
- **Fundamental Machine Learning Concepts**
  - Difference between memorization and generalization
  - What inductive bias is, and what is its role in learning.
  - What underfitting and overfitting means
  - How to take a task and cast it as a learning problem

Why you should never ever touch your test data!!

What we know so far

Decision Trees

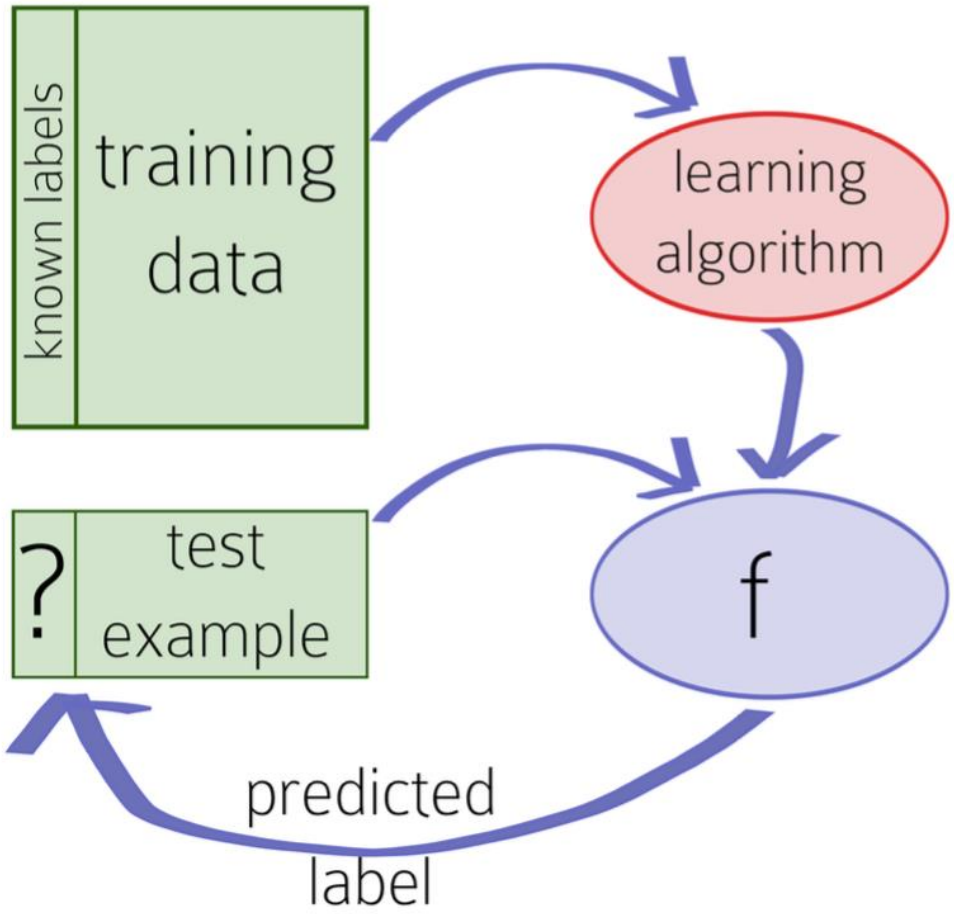
- What is a decision tree, and how to induce it from data

Fundamental Machine Learning Concepts

- Difference between memorization and generalization
- What inductive bias is, and what is its role in learning
- What underfitting and overfitting means
- How to take a task and cast it as a learning problem
- Why you should never ever touch your test data!!

Today's Topics

- Nearest Neighbors (NN) algorithms for classification
  - K-NN, Epsilon ball NN
- Fundamental Machine Learning Concepts
  - Decision boundary


known labels
training data
learning algorithm
?
test example
f
predicted label

Linear Algebra

- Provides compact representation of data
  - For a given example, all its features can be represented as a single **vector**
  - An entire dataset can be represented as a single **matrix**
- Provide ways of manipulating these objects
  - **Dot products**, **vector/matrix operations**, etc
- Provides formal ways of describing and discovering patterns in data
  - Examples are points in a **Vector Space**
  - We can use **Norms and Distances** to compare them
- Some are valid for feature data types
- Some can be made valid, with generalization ...

Mathematical view of vectors

- Ordered set of numbers: (1,2,3,4)
- Example: (*x,y,z*) coordinates of a point in space.
- The 16384 pixels in a 128×128 image of a face
- List of choices in the tennis example
- Vectors usually indicated with bold lower case letters. Scalars with lower case
- Usual mathematical operations with vectors:
  - Addition operation $\boldsymbol{u} + \boldsymbol{v}$, with:
    - Zero $\mathbf{0}$ $\quad \boldsymbol{v} + \mathbf{0} = \boldsymbol{v}$
    - Inverse $-$ $\quad \boldsymbol{v} + (-\boldsymbol{v}) = \mathbf{0}$
  - Scalar multiplication:
    - Distributive rule: $\quad \alpha(\boldsymbol{u} + \boldsymbol{v}) = \alpha\boldsymbol{u} + \alpha\boldsymbol{v}$

      $(\alpha + \beta)\boldsymbol{u} = \alpha\boldsymbol{u} + \beta\boldsymbol{u}$

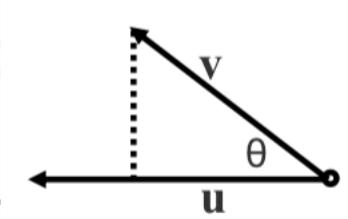
Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

$$\boldsymbol{v}_1 \bullet \boldsymbol{v}_2 = x_1 x_2 + y_1 y_2 + z_1 z_2 \qquad \text{(in 3D)}$$

- Useful for many purposes
  - Computing the Euclidean length of a vector:

    length(**v**) = sqrt(**v• v)**
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors: **u• v**= |**u**| |**v**| cos(θ)
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another
- Other ways of measuring length and distance are possible



Vector Norms

$$\boldsymbol{v} = (x_1, x_2, \ldots, x_n)$$

- Two norm (Euclidean norm)

$$\|\boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

If $\|\boldsymbol{v}\|_2 = 1$, $\boldsymbol{v}$ is a unit vector

- Infinity norm

$$\|\boldsymbol{v}\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|)$$

- One norm (Manhattan distance)

$$\|\boldsymbol{v}\|_1 = \sum_{i=1}^{n} |x_i|$$

Law of Large Numbers

- Suppose that $v_1, v_2, \ldots v_N$ are independent and identically distributed random variables

- The empirical sample average approaches the population average as the number of sample goes to infinity

$$\Pr\left(\lim_{N\to\infty}\frac{1}{N}\sum_{n} v_n = \mathbb{E}[v]\right) = 1$$

Nearest Neighbor

- Intuition–points close in a feature space are likely to belong to the same class
  - Choosing right features is very important
- Nearest Neighbors (NN) algorithms for Classification
  - K-NN, Epsilon ball NN
- Fundamental Machine Learning Concepts
  - Decision boundary

Intuition for Nearest Neighbor Classification

This "rule of nearest neighbor" has considerable elementary intuitive appeal and probably corresponds to practice in many situations. For example, it is possible that much medical diagnosis is influenced by the doctor's recollection of the subsequent history of an earlier patient whose symptoms resemble in some way those of the current patient.

(Fix and Hodges, 1952)

Intuition for Nearest Neighbor Classification

- Simple idea
  - Store all training examples
  - Classify new examples based on label for K closest training examples
  - Training may just involve making structures to make computing closest examples cheaper

K Nearest Neighbor Classifier

K: number of neighbors that classification is based on

Training Data

Test instance with unknown class in $\{-1; +1\}$

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{x}$)

$S \leftarrow [\,]$
for $n = 1$ **to** N **do**
  $S \leftarrow S \oplus \langle \mathrm{d}(\boldsymbol{x}_n, \hat{\boldsymbol{x}}), n \rangle$ // store distance to training example n
end for
$S \leftarrow \text{SORT}(S)$ // put lowest-distance objects first
$\hat{y} \leftarrow 0$
for $k = 1$ **to** K **do**
  $\langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
  $\hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
end for
return $\text{SIGN}(\hat{y})$ // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

2 approaches to learning

Eager learning
(eg decision trees)

- Learn/Train
  - Induce an **abstract model** from data
- Test/Predict/Classify
  - Apply learned model to new data

Lazy learning
(eg nearest neighbors)

- Learn
  - **Just store data** in memory
- Test/Predict/Classify
  - Compare new data to stored data
- Properties
  - Retains all information seen in training
  - Complex hypothesis space
  - Classification can be very slow

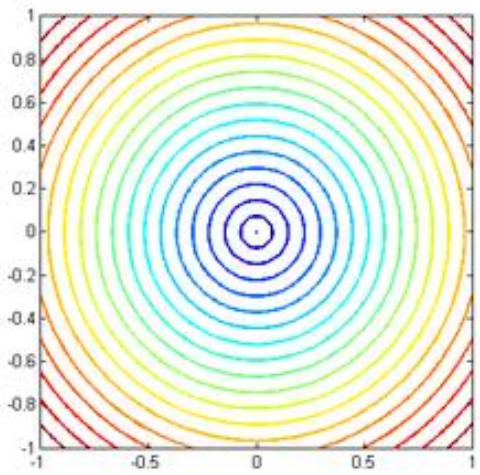
Components of a k-NN Classifier

- Distance metric
  - How do we measure distance between instances?
  - Determines the layout of the example space
- The k hyperparameter
  - How large a neighborhood should we consider?
  - Determines the complexity of the hypothesis space

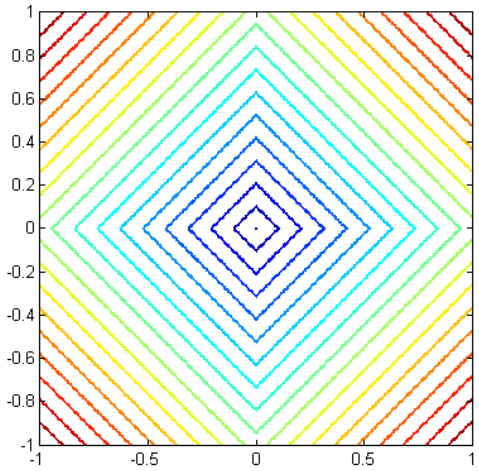
Distance metrics

- We can use any distance function to select nearest neighbors.
- Different distances yield different neighborhoods

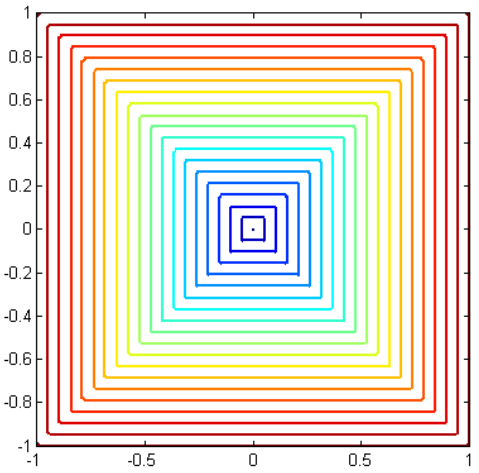
L2 distance
(= Euclidean distance)



L1 distance

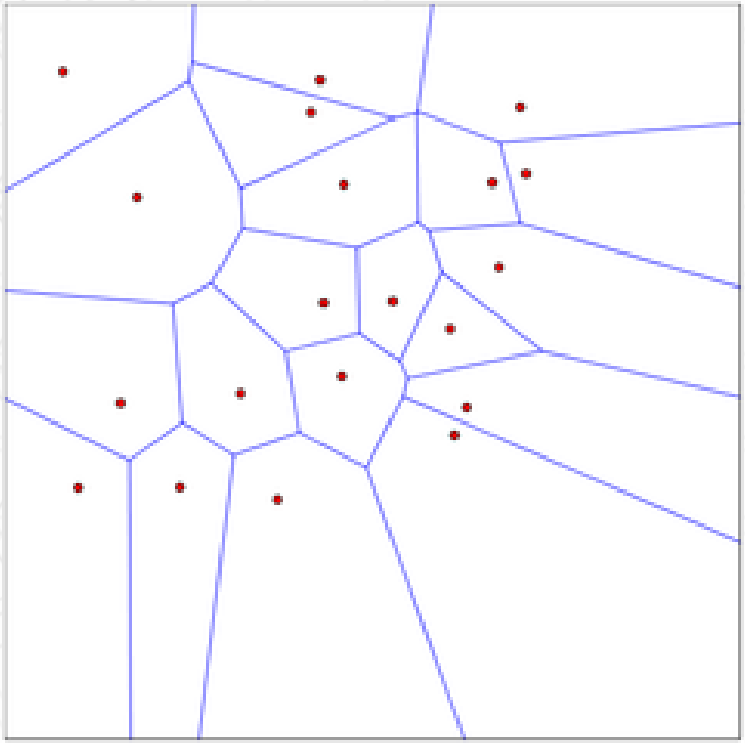


Max norm



K=1 and Voronoi Diagrams

- Imagine we are given a bunch of training examples
- Find regions in the feature space which are closest to every training example
- Algorithm–if our test point is in the region corresponding to a given input point–return its label

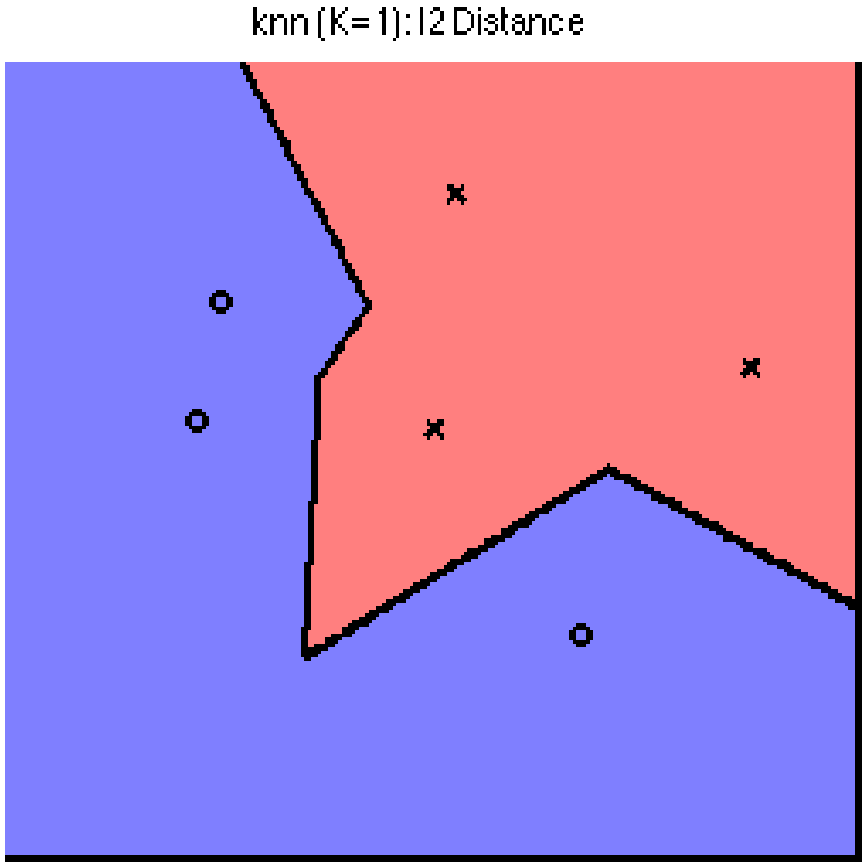


Decision Boundary of a Classifier

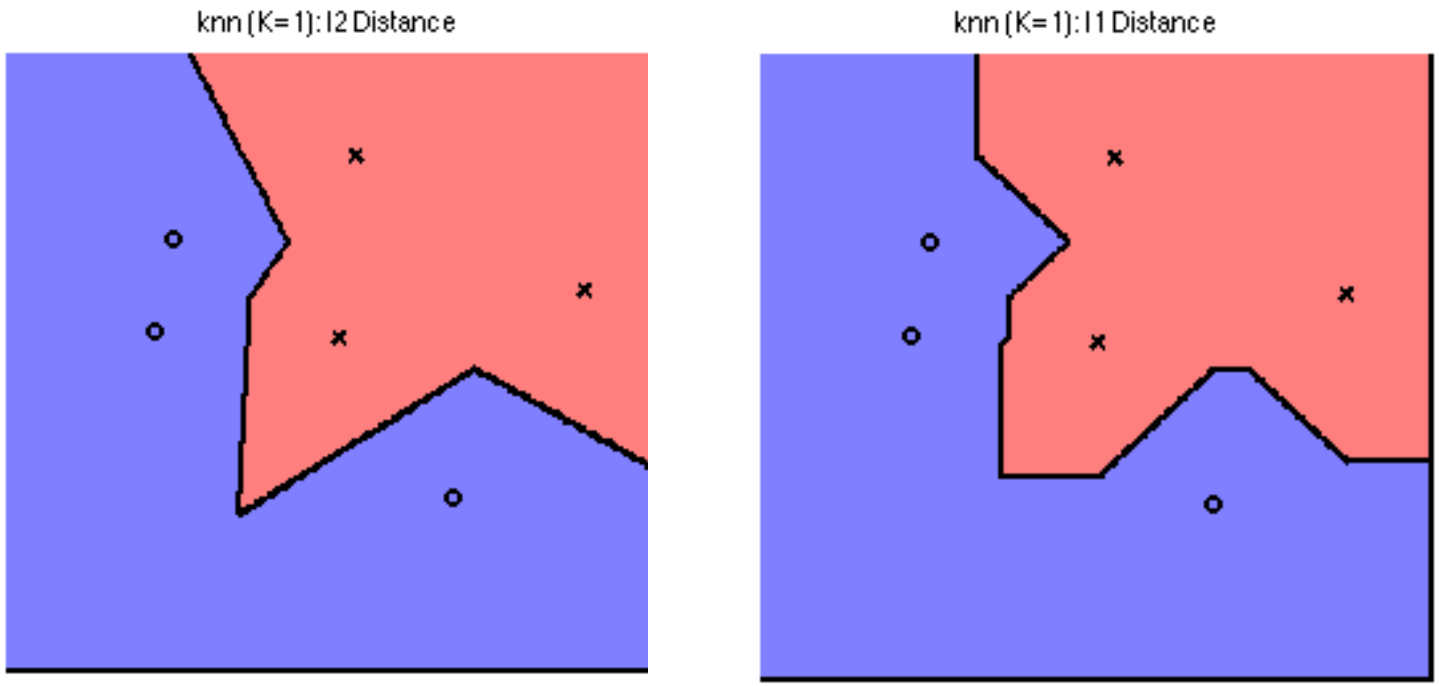
- It is simply the line that separates positive and negative regions in the feature space

- Why is it useful?
  - it helps us visualize how examples will be classified for the entire feature space
  - it helps us visualize the complexity of the learned model

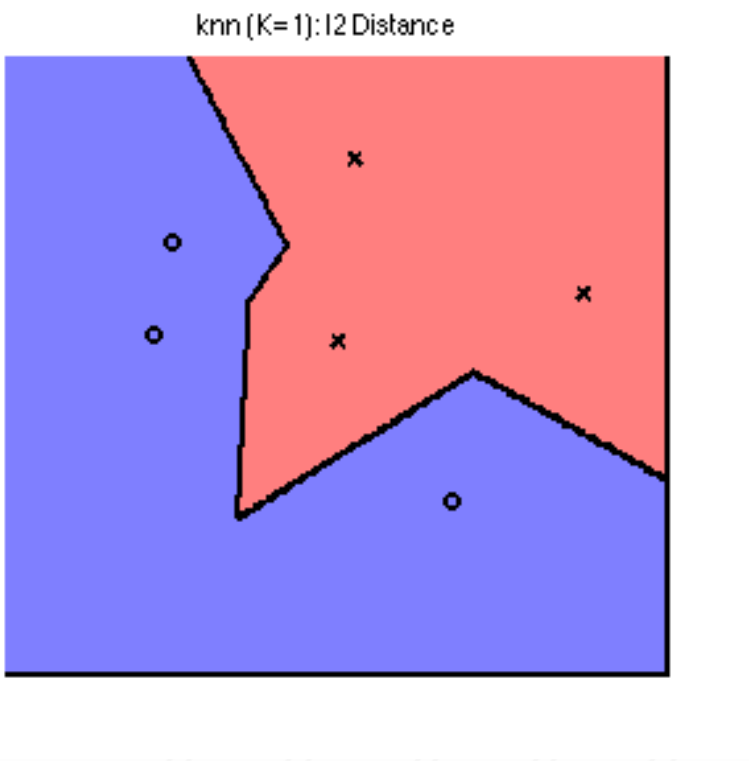
Decision Boundaries for 1-NN

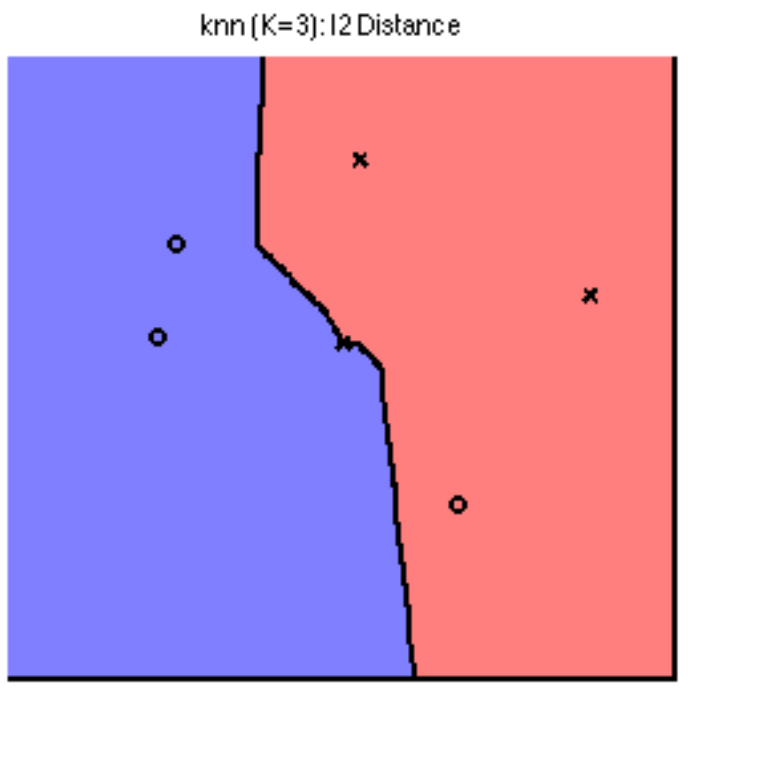


Decision Boundaries change with the distance function



Decision Boundaries change with K





The k hyperparameter

- Tunes the complexity of the hypothesis space
  - If k = 1, every training example has its own neighborhood
  - If k = N, the entire feature space is one neighborhood!
- Higher k yields smoother decision boundaries
- How would you set k in practice?

What is the inductive bias of k-NN?

- Nearby instances should have the same label
- All features are equally important
- Complexity is tuned by the k parameter

Variations on k-NN: Weighted voting

- Weighted voting
  - Default: all neighbors have equal weight
  - Extension: weight neighbors by (inverse) distance

Variations on k-NN: Epsilon Ball Nearest Neighbors

- Same general principle as K-NN, but change the method for selecting which training examples vote

- Instead of using K nearest neighbors, use all examples x such that

$$distance(\hat{x}, x) \leq \varepsilon$$

Exercise: How would you modify KNN-Predict to perform Epsilon Ball NN?

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{\boldsymbol{x}}$)

1: $S \leftarrow [\,]$
2: **for** $n = 1$ **to** N **do**
3: $\quad S \leftarrow S \oplus \langle \mathrm{d}(\boldsymbol{x}_n, \hat{\boldsymbol{x}}), n \rangle$ // store distance to training example n
4: **end for**
5: $S \leftarrow \text{SORT}(S)$ // put lowest-distance objects first
6: $\hat{y} \leftarrow 0$
7: **for** $k = 1$ **to** K **do**
8: $\quad \langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
9: $\quad \hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
10: **end for**
11: **return** SIGN($\hat{y}$) // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

UNIVERSITY OF
MARYLAND

Furong Huang
4124 IRB, College Park, MD 20740
301.405.8010 / furongh@umd.edu