Learning by Computing Distances: Distance-based Methods and Nearest Neighbors

Piyush Rai

Machine Learning (CS771A)

Aug 3, 2016

Data and Data Representation

Data Representation

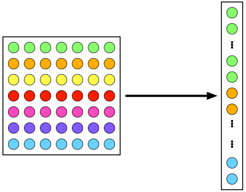
- Learning algorithms work with data given as a set of input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ (supervised), or as a set of inputs $\{\boldsymbol{x}_n\}_{n=1}^N$ (unsupervised)

Data Representation

- Learning algorithms work with data given as a set of input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ (supervised), or as a set of inputs $\{\boldsymbol{x}_n\}_{n=1}^N$ (unsupervised)
- Each $\boldsymbol{x}_n$ is usually a D dimensional feature vector containing the values of D features/attributes that encode properties of the data it represents, e.g.,

Data Representation

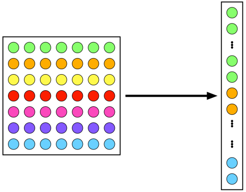
- Learning algorithms work with data given as a set of input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ (supervised), or as a set of inputs $\{\boldsymbol{x}_n\}_{n=1}^N$ (unsupervised)
- Each $\boldsymbol{x}_n$ is usually a D dimensional feature vector containing the values of D features/attributes that encode properties of the data it represents, e.g.,
  - Representing a 7×7 image: $\boldsymbol{x}_n$ can be a 49×1 vector of pixel intensities



  - Note: Good features can also be learned from data (feature learning) or extracted using hand-crafted rules defined by a domain expert

Data Representation

- Learning algorithms work with data given as a set of input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ (supervised), or as a set of inputs $\{\boldsymbol{x}_n\}_{n=1}^N$ (unsupervised)
- Each $\boldsymbol{x}_n$ is usually a D dimensional feature vector containing the values of D features/attributes that encode properties of the data it represents, e.g.,
  - Representing a 7×7 image: $\boldsymbol{x}_n$ can be a 49×1 vector of pixel intensities



  - Note: Good features can also be learned from data (feature learning) or extracted using hand-crafted rules defined by a domain expert
- Output y_n can be real-valued, categorical, or a structured object, depending on the problem (regression, classification, structured output learning, etc.)

Data in Vector Space Representation

- Each feature vector $\boldsymbol{x}_n \in \mathbb{R}^{D\times 1}$ is a point in a D dim. vector space

Data in Vector Space Representation

- Each feature vector $\boldsymbol{x}_n \in \mathbb{R}^{D \times 1}$ is a point in a D dim. vector space
- Standard rules of vector algebra apply on such representations, e.g.,

Data in Vector Space Representation

- Each feature vector $\boldsymbol{x}_n \in \mathbb{R}^{D\times 1}$ is a point in a D dim. vector space
- Standard rules of vector algebra apply on such representations, e.g.,
  - Euclidean distance b/w points $\boldsymbol{x}_n \in \mathbb{R}^D$ and $\boldsymbol{x}_m \in \mathbb{R}^D$

$$d(\boldsymbol{x}_n, \boldsymbol{x}_m) = ||\boldsymbol{x}_n - \boldsymbol{x}_m|| = \sqrt{(\boldsymbol{x}_n - \boldsymbol{x}_m)^\top (\boldsymbol{x}_n - \boldsymbol{x}_m)} = \sqrt{\sum_{d=1}^{D} (x_{nd} - x_{md})^2}$$

Data in Vector Space Representation

- Each feature vector $\boldsymbol{x}_n \in \mathbb{R}^{D\times 1}$ is a point in a D dim. vector space
- Standard rules of vector algebra apply on such representations, e.g.,
    - Euclidean distance b/w points $\boldsymbol{x}_n \in \mathbb{R}^D$ and $\boldsymbol{x}_m \in \mathbb{R}^D$

$$d(\boldsymbol{x}_n, \boldsymbol{x}_m) = ||\boldsymbol{x}_n - \boldsymbol{x}_m|| = \sqrt{(\boldsymbol{x}_n - \boldsymbol{x}_m)^\top (\boldsymbol{x}_n - \boldsymbol{x}_m)} = \sqrt{\sum_{d=1}^{D} (x_{nd} - x_{md})^2}$$

    - Inner-product similarity b/w $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ (cosine, $\boldsymbol{x}_n$, $\boldsymbol{x}_m$ are unit-length vectors)

$$s(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n^\top \boldsymbol{x}_m = \sum_{d=1}^{D} x_{nd} x_{md}$$

Data in Vector Space Representation

- Each feature vector $\boldsymbol{x}_n \in \mathbb{R}^{D\times 1}$ is a point in a D dim. vector space
- Standard rules of vector algebra apply on such representations, e.g.,
  - Euclidean distance b/w points $\boldsymbol{x}_n \in \mathbb{R}^D$ and $\boldsymbol{x}_m \in \mathbb{R}^D$

$$d(\boldsymbol{x}_n, \boldsymbol{x}_m) = ||\boldsymbol{x}_n - \boldsymbol{x}_m|| = \sqrt{(\boldsymbol{x}_n - \boldsymbol{x}_m)^\top (\boldsymbol{x}_n - \boldsymbol{x}_m)} = \sqrt{\sum_{d=1}^{D} (x_{nd} - x_{md})^2}$$

  - Inner-product similarity b/w $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ (cosine, $\boldsymbol{x}_n$, $\boldsymbol{x}_m$ are unit-length vectors)

$$s(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n^\top \boldsymbol{x}_m = \sum_{d=1}^{D} x_{nd} x_{md}$$

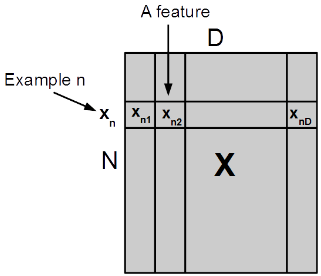
  - ℓ_1 distance between two points $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$

$$d_1(\boldsymbol{x}_n, \boldsymbol{x}_m) = ||\boldsymbol{x}_n - \boldsymbol{x}_m||_1 = \sum_{d=1}^{D} |x_{nd} - x_{md}|$$

More on Data Representation..

We will (usually) assume that:

- $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ denotes data in form of an $N \times D$ feature matrix
- N examples, D features to represent each example
- Each row is an example, each column is a feature
- $\boldsymbol{x}_n$ denotes the n-th example (a vector of length D)

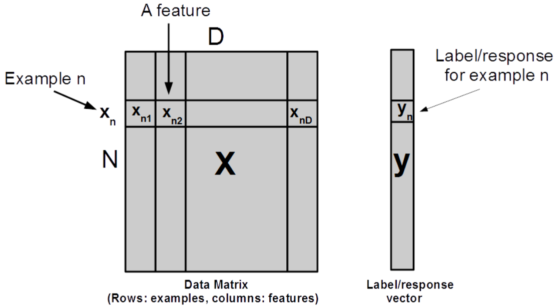


Data Matrix
(Rows: examples, columns: features)

More on Data Representation..

We will (usually) assume that:

- $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ denotes data in form of an $N \times D$ feature matrix
- N examples, D features to represent each example
- Each row is an example, each column is a feature
- $\boldsymbol{x}_n$ denotes the n-th example (a vector of length D)

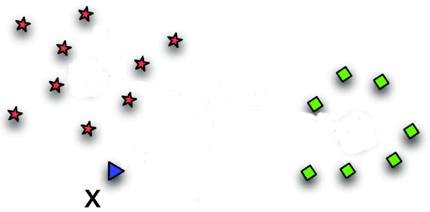


- $\boldsymbol{y}$ denotes labels/responses in form of an $N \times 1$ label/response vector
- y_n denotes label/response of the n-th example $\boldsymbol{x}_n$

Our First Learning Algorithm

A Simple Distance-based Classifier

- Given: N labeled training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ from two classes
    - Assume green is positive and red is negative class
    - N_+ exampes from positive class, N_- examples from negative class
- Our goal: Learn a model to predict label (class) y for a new test example $\boldsymbol{x}$



Pic credit: Modified from the original by Alex Smola and Vishy Vishwanathan

A Simple Distance-based Classifier

- Given: N labeled training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ from two classes
  - Assume green is positive and red is negative class
  - N_+ exampes from positive class, N_- examples from negative class
- Our goal: Learn a model to predict label (class) y for a new test example $\boldsymbol{x}$

- A simple "distance from means" model: assign to class with closer mean

Pic credit: Modified from the original by Alex Smola and Vishy Vishwanathan

A Simple Distance-based Classifier

- Given: N labeled training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ from two classes
  - Assume green is positive and red is negative class
  - N_+ exampes from positive class, N_- examples from negative class
- Our goal: Learn a model to predict label (class) y for a new test example $\boldsymbol{x}$

- A simple "distance from means" model: assign to class with closer mean
- Note: The basic idea generalizes to more than 2 classes as well

Pic credit: Modified from the original by Alex Smola and Vishy Vishwanathan

Distance from Means: More Formally

- What does the decision rule look like, mathematically ?

Distance from Means: More Formally

- What does the decision rule look like, mathematically ?

- The mean of each class is given by

$$\mu_- = \frac{1}{N_-}\sum_{y_n=-1} \boldsymbol{x}_n \quad \text{and} \quad \mu_+ = \frac{1}{N_+}\sum_{y_n=+1} \boldsymbol{x}_n$$

Distance from Means: More Formally

- What does the decision rule look like, mathematically ?

- The mean of each class is given by

$$\boldsymbol{\mu}_- = \frac{1}{N_-} \sum_{y_n=-1} \boldsymbol{x}_n \quad \text{and} \quad \boldsymbol{\mu}_+ = \frac{1}{N_+} \sum_{y_n=+1} \boldsymbol{x}_n$$

- Note: We can simply store the two means and throw away the training data

Distance from Means: More Formally

- Distances from each mean are given by

$$||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 = ||\boldsymbol{\mu}_-||^2 + ||\boldsymbol{x}||^2 - 2\langle \boldsymbol{\mu}_-, \boldsymbol{x}\rangle$$
$$||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = ||\boldsymbol{\mu}_+||^2 + ||\boldsymbol{x}||^2 - 2\langle \boldsymbol{\mu}_+, \boldsymbol{x}\rangle$$

- Note: $||\boldsymbol{a} - \boldsymbol{b}||^2$ denotes squared Euclidean distance b/w two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
- Note: $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes inner product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
- Note: $||\boldsymbol{a}||^2 = \langle \boldsymbol{a}, \boldsymbol{a}\rangle$ denotes the squared ℓ_2 norm of $\boldsymbol{a}$

Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x}\rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x}\rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

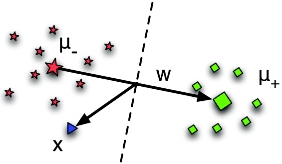
- If $f(\boldsymbol{x}) > 0$ then class is $+1$, otherwise class is -1. Basically $y = \text{sign}[f(\boldsymbol{x})]$

Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x}\rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

- If $f(\boldsymbol{x}) > 0$ then class is $+1$, otherwise class is -1. Basically $y = \text{sign}[f(\boldsymbol{x})]$
- **Imp.:** $f(\boldsymbol{x})$ effectively denotes a hyperplane based classification rule (with the vector $\boldsymbol{w} = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-$ representing the direction normal to the hyperplane)

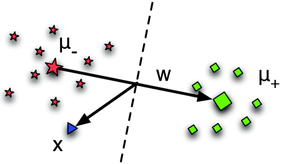


Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x}\rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

- If $f(\boldsymbol{x}) > 0$ then class is $+1$, otherwise class is -1. Basically $y = \text{sign}[f(\boldsymbol{x})]$
- **Imp.:** $f(\boldsymbol{x})$ effectively denotes a hyperplane based classification rule (with the vector $\boldsymbol{w} = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-$ representing the direction normal to the hyperplane)



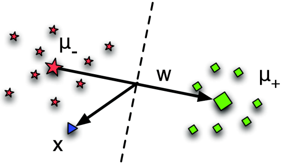
- **Imp.:** Can show that the rule is equivalent to $f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x}\rangle + b$, where α's and b can be estimated from training data (homework problem)

Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x}\rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

- If $f(\boldsymbol{x}) > 0$ then class is $+1$, otherwise class is -1. Basically $y = \text{sign}[f(\boldsymbol{x})]$
- **Imp.:** $f(\boldsymbol{x})$ effectively denotes a hyperplane based classification rule (with the vector $\boldsymbol{w} = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-$ representing the direction normal to the hyperplane)



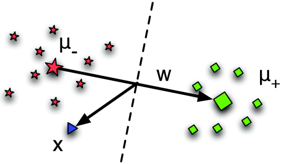
- **Imp.:** Can show that the rule is equivalent to $f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x}\rangle + b$, where α's and b can be estimated from training data (homework problem)
    - This specific form of the decision rule is very important. Decision rules for many (in fact most) supervised learning algorithms can be written like this

Distance from Means: The Decision Rule

- Let us denote by $f(\boldsymbol{x})$ our decision rule, defined as

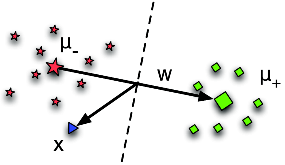
$$f(\boldsymbol{x}) := ||\boldsymbol{\mu}_- - \boldsymbol{x}||^2 - ||\boldsymbol{\mu}_+ - \boldsymbol{x}||^2 = 2\langle \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \boldsymbol{x} \rangle + ||\boldsymbol{\mu}_-||^2 - ||\boldsymbol{\mu}_+||^2$$

- If $f(\boldsymbol{x}) > 0$ then class is $+1$, otherwise class is -1. Basically $y = \text{sign}[f(\boldsymbol{x})]$
- **Imp.:** $f(\boldsymbol{x})$ effectively denotes a hyperplane based classification rule (with the vector $\boldsymbol{w} = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-$ representing the direction normal to the hyperplane)



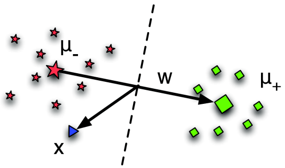
- **Imp.:** Can show that the rule is equivalent to $f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle + b$, where α's and b can be estimated from training data (homework problem)
  - This specific form of the decision rule is very important. Decision rules for many (in fact most) supervised learning algorithms can be written like this
  - The inner product above can be replaced by more general similarity functions

Some aspects of "Distance from Means"



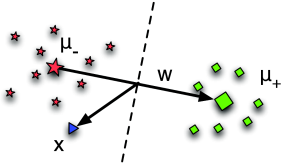
- Simple to understand and implement

Some aspects of "Distance from Means"



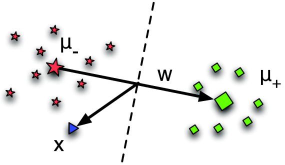
- Simple to understand and implement
- Would usually require plenty of training data for each class (to estimate the mean of each class reliably)

Some aspects of “Distance from Means”



- Simple to understand and implement
- Would usually require plenty of training data for each class (to estimate the mean of each class reliably)
- Can only learn linear decision boundaries, unless Euclidean distance is replaced by a nonlinear distance function (more when we talk about Kernels)

Some aspects of "Distance from Means"

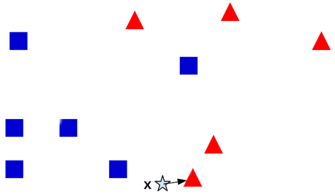


- Simple to understand and implement
- Would usually require plenty of training data for each class (to estimate the mean of each class reliably)
- Can only learn linear decision boundaries, unless Euclidean distance is replaced by a nonlinear distance function (more when we talk about Kernels)
- Can be made more rigorous by modeling each class by a more sophisticated class probability distribution (e.g., a multivariate Gaussian) and computing distances of test points from these class distributions (rathar than means)
  - Several methods have this flavor, e.g., Linear (Fisher) Discriminant Analysis

Nearest Neighbor Methods

Nearest Neighbor

- Another classic distance-based supervised learning method

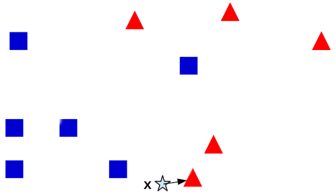


- The label y for $\boldsymbol{x} \in \mathbb{R}^D$ will be the label of its nearest neighbor in training data. Also known as one-nearest-neighbor (1-NN)
- Euclidean distance can be used to find the nearest neighbor

[1] Distance Metric Learning. See "A Survey on Metric Learning for Feature Vectors and Structured Data" by Ballet *et al*

Nearest Neighbor

- Another classic distance-based supervised learning method



- The label y for $\boldsymbol{x} \in \mathbb{R}^D$ will be the label of its nearest neighbor in training data. Also known as one-nearest-neighbor (1-NN)
- Euclidean distance can be used to find the nearest neighbor
- The Euclidean distance can also be replaced by other distances metrics, e.g.,

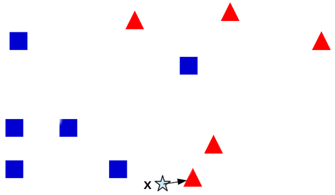
$$d_M(\boldsymbol{x}_n, \boldsymbol{x}_m) = \sqrt{(\boldsymbol{x}_n - \boldsymbol{x}_m)^\top \mathbf{M}(\boldsymbol{x}_n - \boldsymbol{x}_m)}$$

where $\mathbf{M}$ is a $D \times D$ a p.s.d. matrix (can be learned[1] from data)

[1] Distance Metric Learning. See "A Survey on Metric Learning for Feature Vectors and Structured Data" by Ballet *et al*

Nearest Neighbor

- Another classic distance-based supervised learning method



- The label y for $\boldsymbol{x} \in \mathbb{R}^D$ will be the label of its nearest neighbor in training data. Also known as one-nearest-neighbor (1-NN)
- Euclidean distance can be used to find the nearest neighbor
- The Euclidean distance can also be replaced by other distances metrics, e.g.,

$$d_M(\boldsymbol{x}_n, \boldsymbol{x}_m) = \sqrt{(\boldsymbol{x}_n - \boldsymbol{x}_m)^\top \mathbf{M}(\boldsymbol{x}_n - \boldsymbol{x}_m)}$$

  where $\mathbf{M}$ is a $D \times D$ a p.s.d. matrix (can be learned[1] from data)
- Note: The method also applies to regression problems (real-valued labels)

[1] Distance Metric Learning. See "A Survey on Metric Learning for Feature Vectors and Structured Data" by Ballet *et al*

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)
- Predction can be slow at test time

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)
- Predction can be slow at test time
  - For each test point, need to compute its distance from all the training points

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)
- Predction can be slow at test time
  - For each test point, need to compute its distance from all the training points
  - Clever data-structures or data-summarization techniques can provide speed-ups

Some Aspects about Nearest Neighbor

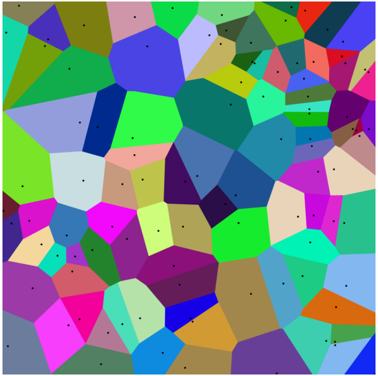
- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the “Bayes optimal” classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No “model” is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)
- Predction can be slow at test time
  - For each test point, need to compute its distance from all the training points
  - Clever data-structures or data-summarization techniques can provide speed-ups
- Need to be careful in choosing the distance function to compute distances (especially when the data dimension D is very large)

Some Aspects about Nearest Neighbor

- A simple yet very effective method in practice (if given lots of training data)
  - *Provably* has an error-rate that is no worse than twice of the "Bayes optimal" classifier which assumes knowledge of the true data distribution
- Also called a memory-based or instance-based or non-parametric method
- No "model" is learned here. Prediction step uses all the training data
- Requires lots of storage (need to keep all the training data at test time)
- Predction can be slow at test time
  - For each test point, need to compute its distance from all the training points
  - Clever data-structures or data-summarization techniques can provide speed-ups
- Need to be careful in choosing the distance function to compute distances (especially when the data dimension D is very large)
- The 1-NN can suffer if data contains outliers (we will soon see a geometric illustration), or if amount of training data is small. Using more neighbors ($K > 1$) is usually more robust

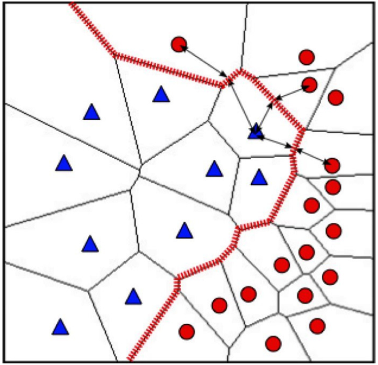
Geometry of 1-NN

- 1-NN induces a Voronoi tessellation of the input space



The Decision Boundary of 1-NN

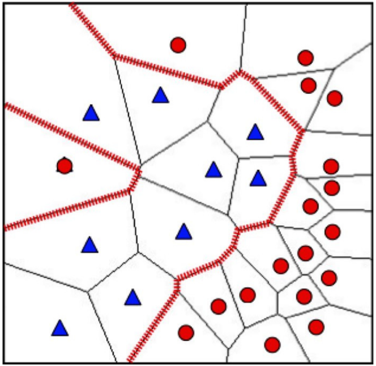
- The decision boundary is composed of hyperplanes that form perpendicular bisectors of pairs of points from different classes



Pic credit: Victor Lavrenko

Effect of Outliers on 1-NN

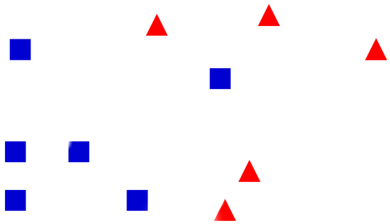
- An illustration of how the decision boundary can drastically change when the data contains some outliers



Pic credit: Victor Lavrenko

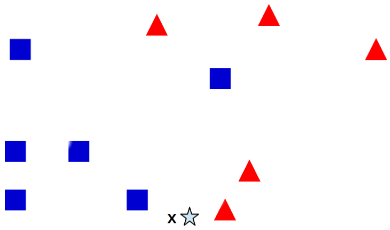
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



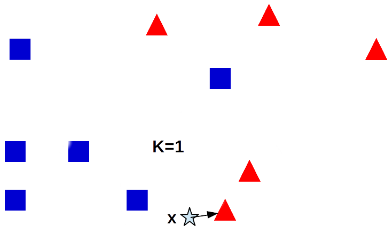
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



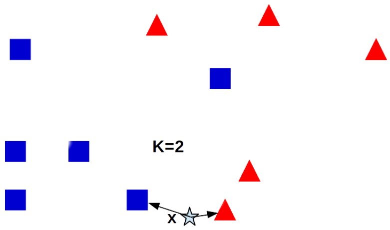
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



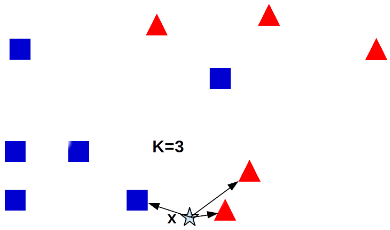
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



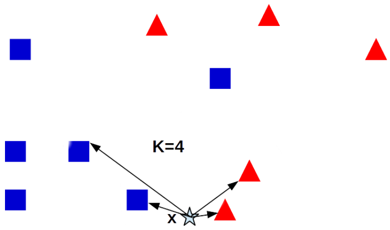
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



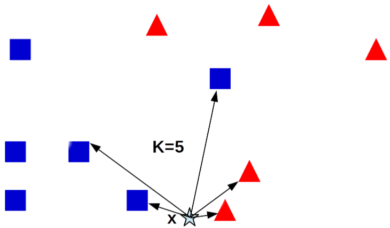
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



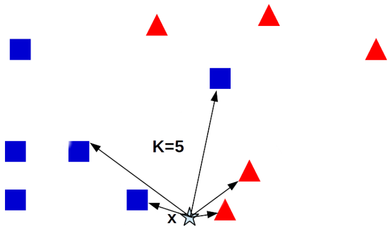
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



K-Nearest Neighbors (K-NN)

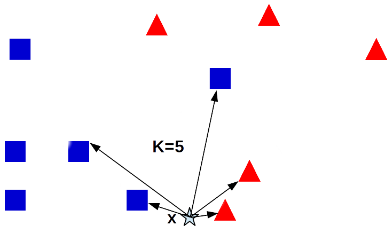
- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



- Works for both classification and regression

K-Nearest Neighbors (K-NN)

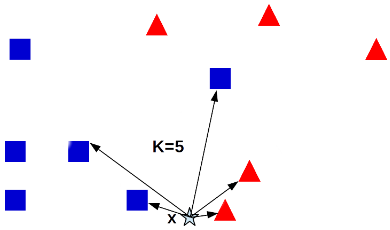
- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



- Works for both classification and regression
  - For classification, we usually take the majority labels from the K neighbors

K-Nearest Neighbors (K-NN)

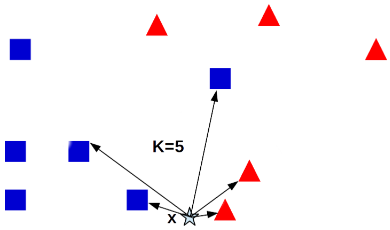
- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



- Works for both classification and regression
  - For classification, we usually take the majority labels from the K neighbors
  - For regression, we usually average the real-valued labels of the K neighbors

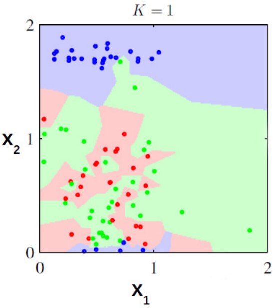
K-Nearest Neighbors (K-NN)

- Makes one-nearest-neighbor more robust by using more than one neighbor
- The K-NN prediction rule: Take a majority vote (or average) of the labels of $K > 1$ neighbors in the training data



- Works for both classification and regression
  - For classification, we usually take the majority labels from the K neighbors
  - For regression, we usually average the real-valued labels of the K neighbors
- The "right" value of K needs to be selected (e.g., via cross-validation)

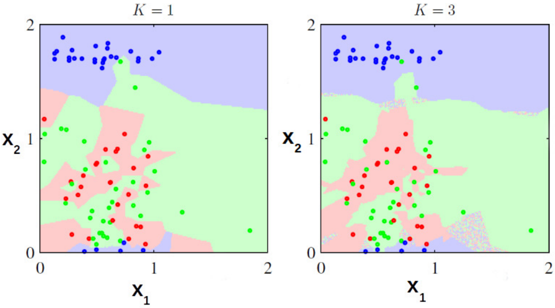
- Larger K leads to smoother decision boundaries



Pic credit: Chris Bishop (PRML)

K-Nearest Neighbors: Decision Boundaries

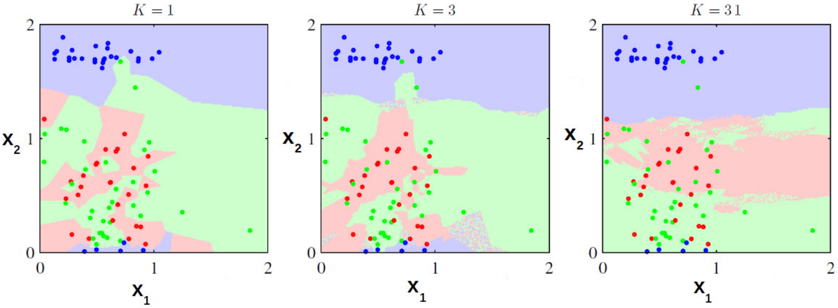
- Larger K leads to smoother decision boundaries



Pic credit: Chris Bishop (PRML)

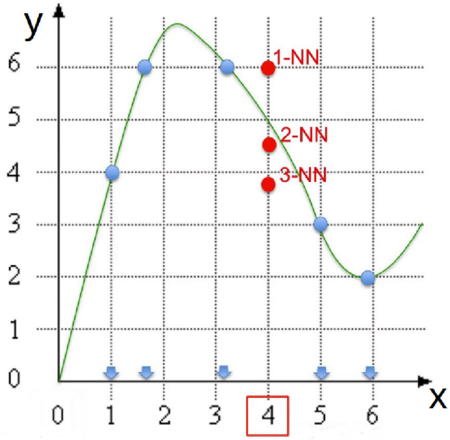
K-Nearest Neighbors: Decision Boundaries

- Larger K leads to smoother decision boundaries



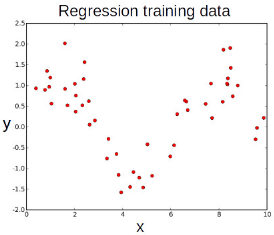
Pic credit: Chris Bishop (PRML)

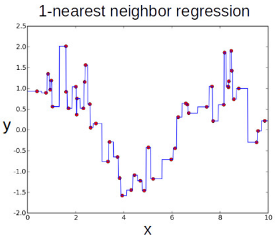
K-NN Behavior for Regression

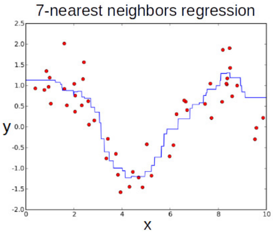


Pic credit: Victor Lavrenko

K-NN Behavior for Regression







Pic credit: Alex Smola and Vishy Vishwanathan

Summary

- Looked at two distance-based methods for classification/regression
  - A "Distance from Means" Method
  - Nearest Neighbors Method

Summary

- Looked at two distance-based methods for classification/regression
  - A "Distance from Means" Method
  - Nearest Neighbors Method
- Both are simple to understand and only require knowledge of basic geometry

Summary

- Looked at two distance-based methods for classification/regression
  - A "Distance from Means" Method
  - Nearest Neighbors Method
- Both are simple to understand and only require knowledge of basic geometry
- Have connections to other more advanced methods (as we will see)

Summary

- Looked at two distance-based methods for classification/regression
  - A "Distance from Means" Method
  - Nearest Neighbors Method
- Both are simple to understand and only require knowledge of basic geometry
- Have connections to other more advanced methods (as we will see)
- Need to be careful when computing the distances (or when deciding which distance function to use). Euclidean distance may not work well in many cases, especially in high dimensions (or may have to do some careful scaling of features if different features are not on the same scale)

Next Class:
Decision Trees for Classification and Regression