Classification with Handwritten Digits

−*Classes without Quizzes*, An SJSU Alumni Association Event

Dr. Guangliang Chen

Dept. of Math & Statistics, College of Science, SJSU

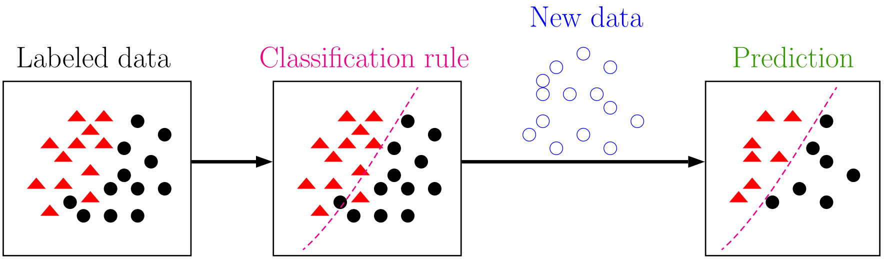
April 23, 2016

Outline

- Introduction to classification
- Description of the dataset
- Classification algorithms
  - Instance-based classifiers
  - Support vector machines (SVM)
  - Classification trees and ensemble learning
- Summary

What is classification?

Classification is a *machine learning* problem about how to assign labels to new data based on a given set of labeled data.



What is classification? (cont'd)

Some terminology:

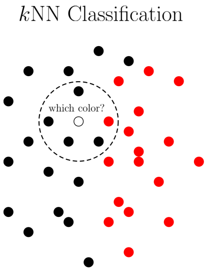
- Labeled data is called **training data**;
- New data is called **test data**;
- A classification rule/algorithm is called a **classifier**.

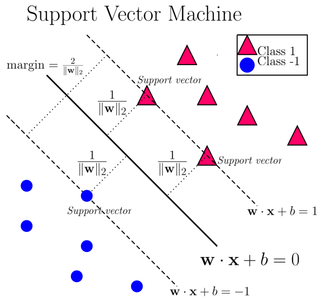
Classification has numerous applications, e.g., spam email detection, digit recognition, face recognition, and document classification.

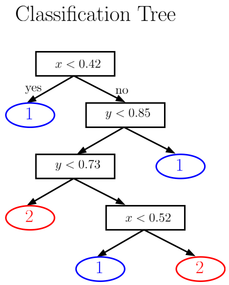
Lots of algorithms have been developed, leading to a vast literature on classification.

Classifiers to be introduced later

In this presentation we will focus on the following three methods because (1) they represent different kinds of methods and (2) they often have excellent performance.

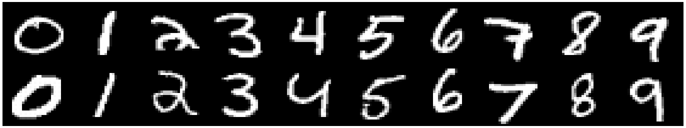



Support Vector Machine
Class 1
Class -1
margin $= \frac{2}{\|\mathbf{w}\|_2}$
$\frac{1}{\|\mathbf{w}\|_2}$
Support vector
Support vector
Support vector
$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$



Handwritten digits recognition

We teach classification mainly based on the the digit recognition problem: Given a set of training examples



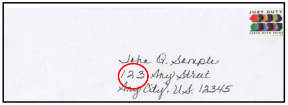
determine what digits the test images contain by machine:

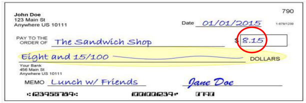
Why digit recognition?

- Simple, intuitive to understand
- Practically important

Potential Applications

- **Banking**: Check deposits
- **Surveillance**: license plates
- **Shipping**: Envelopes/Packages


John Q. Sample
123 Any Street
Any City, U.S. 12345


790

John Doe
123 Main St
Anywhere US 10111

Date 01/01/2015 1-678/1239

PAY TO THE ORDER OF The Sandwich Shop $ 8.15

Eight and 15/100 DOLLARS

Your Bank
456 Main St
Anywhere US 10111

MEMO Lunch w/ Friends Jane Doe

⑆123456789⑆ 1001001234⑈ 0790

Our main data set: MNIST handwritten digits

The MNIST database of handwritten digits, formed by Yann LeCun of NYU, has a total of 70,000 examples from approximately 250 writers:

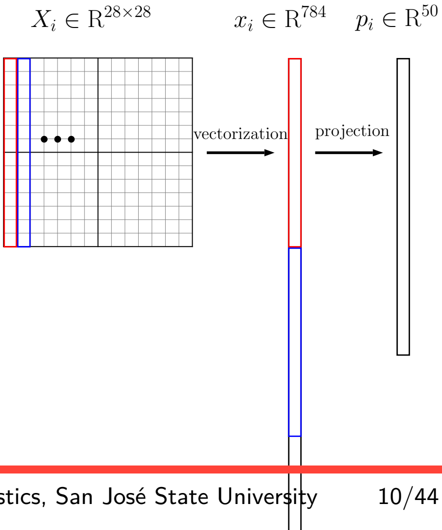
- The images are 28×28 in size
- The training set contains 60,000 images while the test set has 10,000
- It is a benchmark dataset used by many people

Why MNIST?

- Easy to use, yet difficult enough for classification
  - Big data (large size and high dimensionality)
  - 10 classes $(0, 1, \ldots, 9)$
  - Great variability (due to different ways people write)
  - Nonlinear separation between the classes
- Well studied (thus lots of learning resources available):
  - It is used by an ongoing Kaggle competition

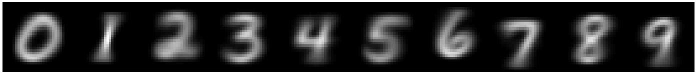
Preprocessing the digits

- The original format is matrix (of size 28×28);
- They can be converted to vectors (784 dimensional), required by most algorithms.
- Due to high dimensionality, we project the data into 50 dimensions (while preserving most variance)

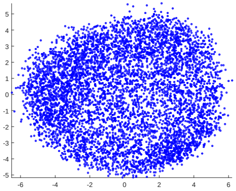

$X_i \in \mathrm{R}^{28\times 28}$
$x_i \in \mathrm{R}^{784}$
$p_i \in \mathrm{R}^{50}$
vectorization
projection

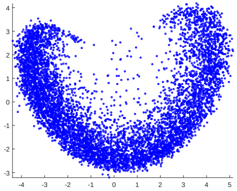
Visualization of the data set

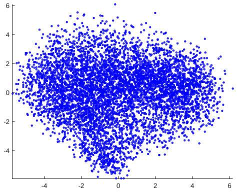
1. The “average” writer



2. The full appearance of the digit clouds 0, 1, 2 respectively



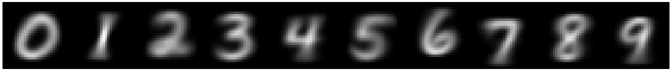




A very first attempt at classification

Naive approach: Assign labels to test images

based on the most similar class centers:

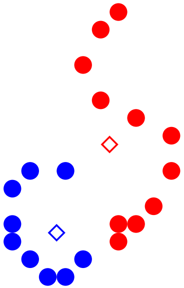


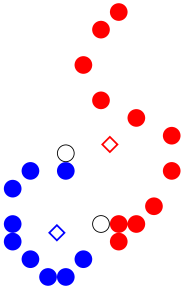
We call this method **nearest centroid** classifier.

It has **18.0% error rate**, i.e., 1,800 errors (out of 10,000).

Why is nearest centroid not so good?

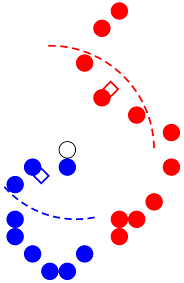
The reason is that the class centroids are often insufficient to represent the respective classes (due to their complex global structures):





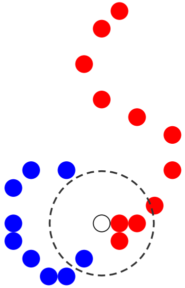
How to fix this issue?

One solution is to use centroids of nearest subsets of the classes (of some fixed size k). We call this method **nearest local centroid**.

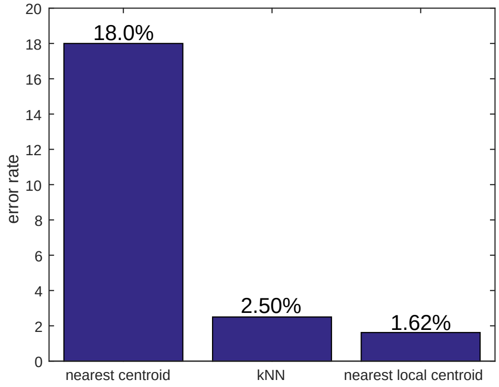


How to fix this issue?

A second solution is to use majority vote among the k nearest neighbors (from all classes). This method is often called k **nearest neighbors (**k**NN)**.



Methods comparison

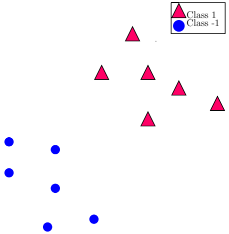


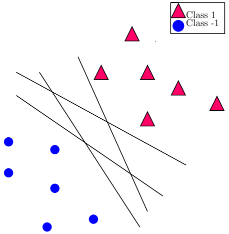
Comments on nearest local centroid and kNN

- Model free (nonparametric)
- Lazy learning
- Simple to implement, yet quite powerful
- Algorithmic complexity only depends nearest neighbors search (i.e., memory and CPU cost)
- The choice of k is critical (often selected by cross validation)
- Lots of variations (weighted kNN, different metrics)

Support Vector Machine (SVM)

To introduce the idea of SVM, we consider binary classification first.



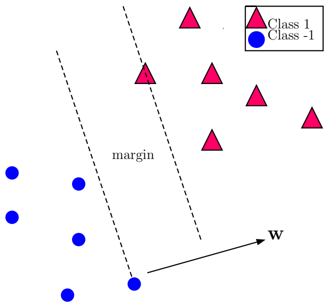


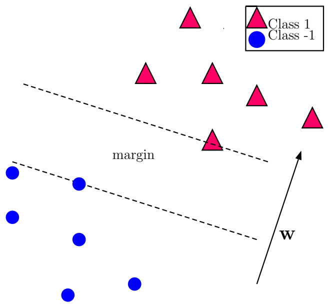
SVM effectively compares (under some criterion) all hyperplanes $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is a normal vector while b determines location.

Classification with Handwritten Digits

The following specifies how to logically think about the problem:

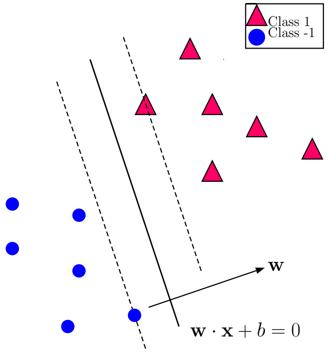
- Any fixed direction $\mathbf{w}$ determines a unique margin.





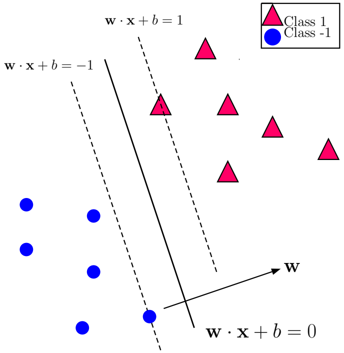
Classification with Handwritten Digits

- We select b such that the center hyperplane is given by $\mathbf{w}\cdot\mathbf{x}+b=0$. This is the **optimal** boundary *orthogonal to the given direction* $\mathbf{w}$, as it balances the two classes.



Classification with Handwritten Digits

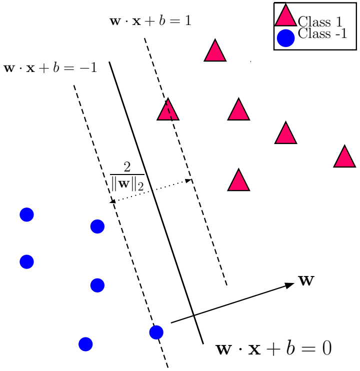
- Any scalar multiple of $\mathbf{w}$ and b denotes the same hyperplane. To uniquely fix the two parameters, we require the margin boundaries to have equations $\mathbf{w} \cdot \mathbf{x} + b = \pm 1$.



$\mathbf{w} \cdot \mathbf{x} + b = 1$
$\mathbf{w} \cdot \mathbf{x} + b = -1$
Class 1
Class -1
$\mathbf{w}$
$\mathbf{w} \cdot \mathbf{x} + b = 0$

Classification with Handwritten Digits

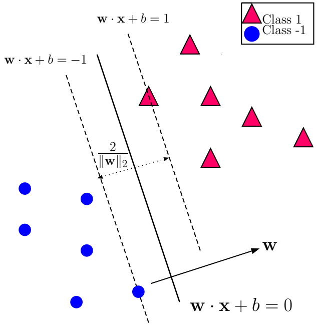
- Under such requirements, we can show that the margin between the two classes is exactly $\frac{2}{\|\mathbf{w}\|_2}$.


$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
Class 1
Class -1
$\frac{2}{\|\mathbf{w}\|_2}$
$\mathbf{w}$
$\mathbf{w}\cdot\mathbf{x}+b=0$

The binary SVM problem

Problem. Given training data $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ with labels $y_i = \pm 1$, SVM finds the optimal separating hyperplane by maximizing the class margin.

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|_2} \quad \text{subject to}$$
$$\mathbf{w}\cdot\mathbf{x}_i + b \ge 1, \quad \text{if } y_i = +1;$$
$$\mathbf{w}\cdot\mathbf{x}_i + b \le -1, \quad \text{if } y_i = -1$$


$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
Class 1
Class -1
$\frac{2}{\|\mathbf{w}\|_2}$
$\mathbf{w}$
$\mathbf{w}\cdot\mathbf{x}+b=0$

A more convenient formulation

The previous problem can be reformulated as follows:

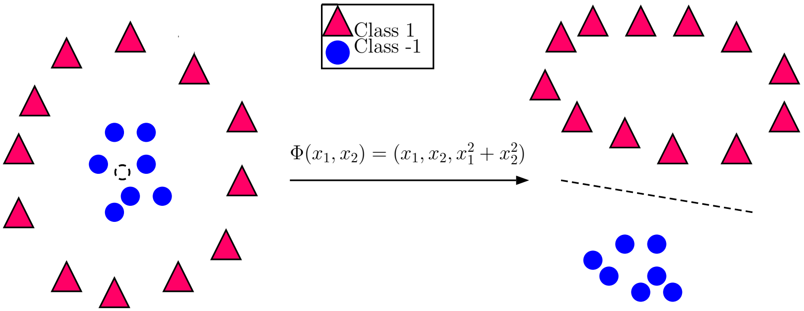
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 \qquad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \text{ for all } 1 \leq i \leq n.$$

Remarks:

- This problem can be efficiently solved by **quadratic programming**.
- The optimal $\mathbf{w}$ is *a linear combination of the support vectors* (b also only depends on the support vectors).

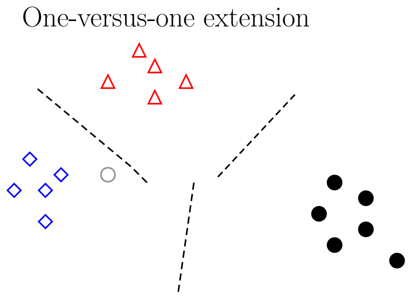
What if the classes are nonlinearly separated?

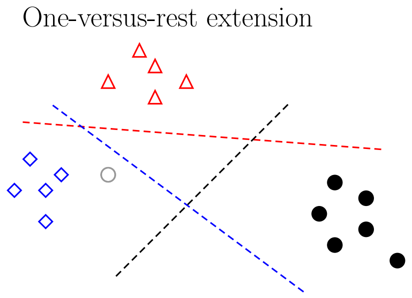
Strategy: map given data to a different space (called feature space):



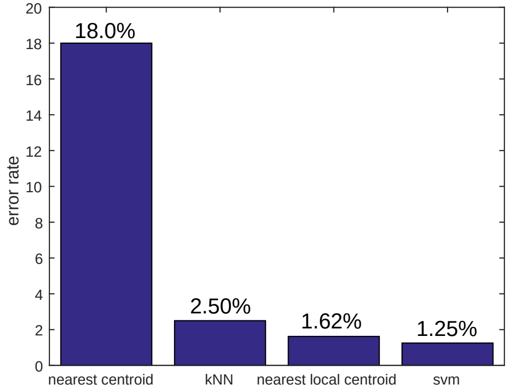
What if there are more than 2 classes?

Binary SVM can be extended to a multiclass setting in the following ways:





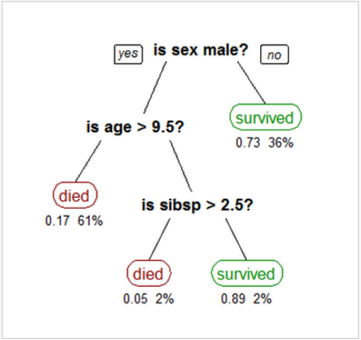
Methods comparison (including SVM)



Comments on SVM

- Invented by Vapnik in the 1990's
- Based on nice theory
- Excellent generalization properties
- Globally optimal solution
- Can handle outliers and nonlinear boundary simultaneously
- One of the most important developments in machine learning

A classification tree

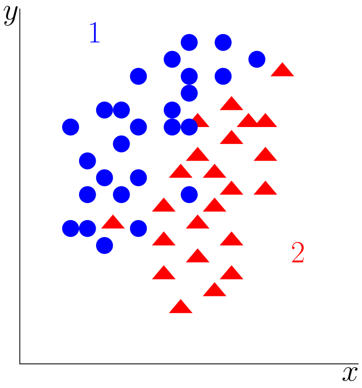


A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

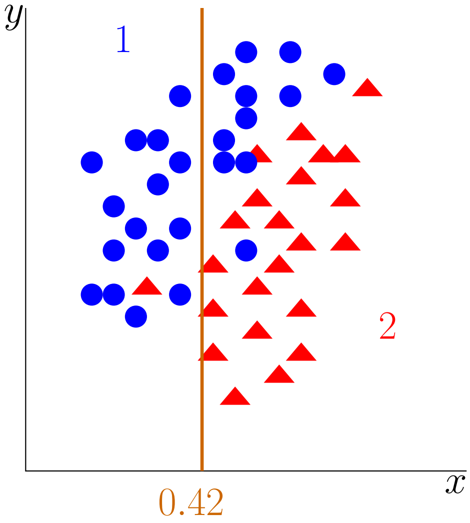
(1) The tree is grown using training data, by recursive splitting.
(2) Each internal node represents a query on one of the variables.
(3) The terminal nodes are the decision nodes, typically dominated by one of the classes.
(4) New observations are classified in the respective terminal nodes by using majority vote.

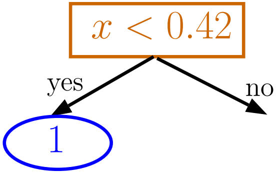
Demonstration: how to build a classification tree

Consider the following training data.

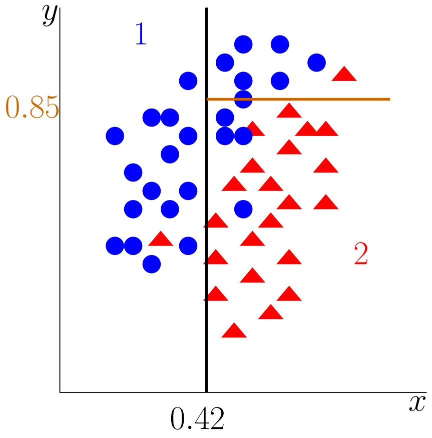


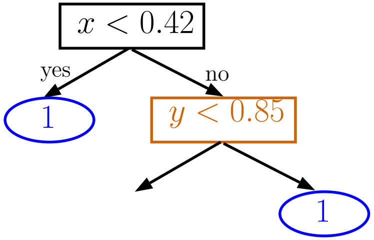
Classification with Handwritten Digits



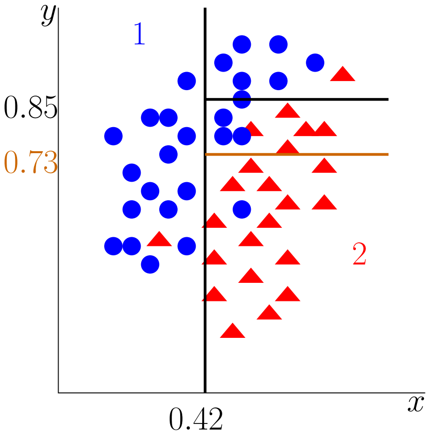


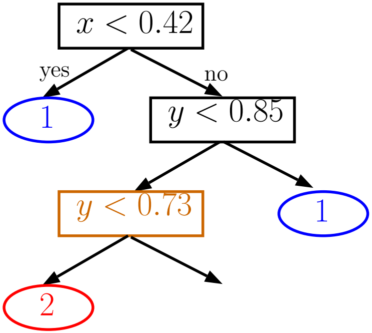
Classification with Handwritten Digits



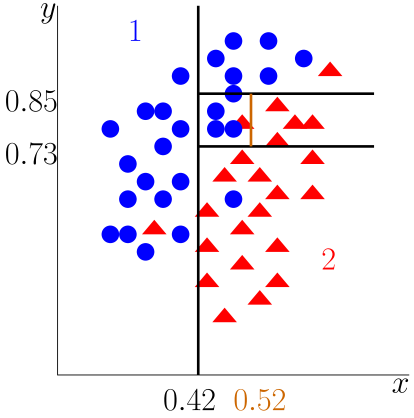


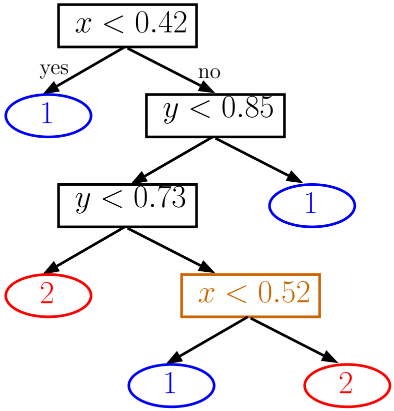
Classification with Handwritten Digits



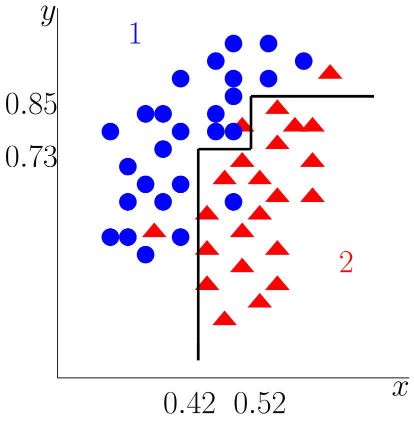


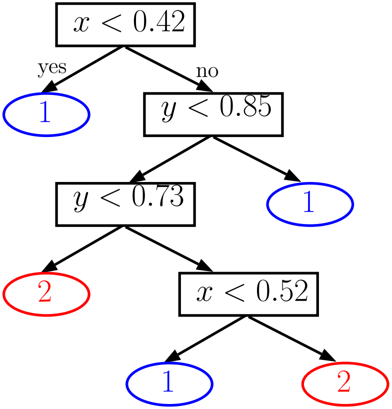
Classification with Handwritten Digits



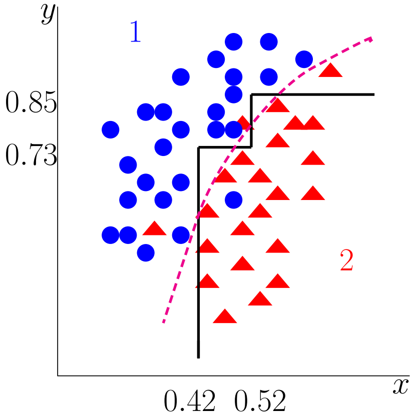


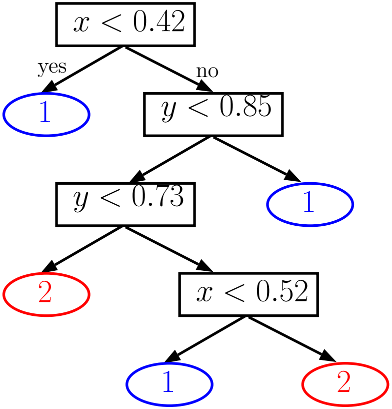
Classification with Handwritten Digits





Classification with Handwritten Digits





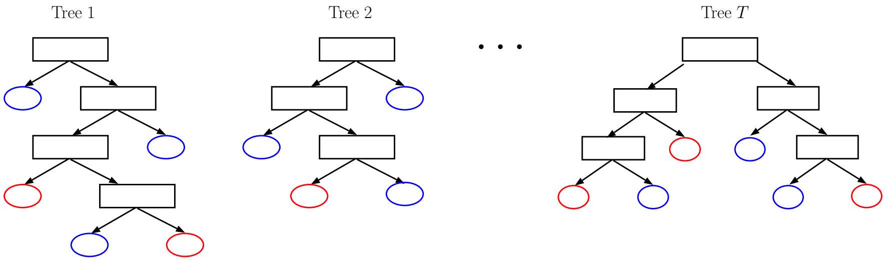
Remark about classification trees

Classification trees have lots of advantages (e.g., simple and fast to train), but are considered as weak learners for the following reasons:

- The decision boundary is piecewise linear (consisting of only horizontal and vertical lines)
- Unstable (if we change the data a little, the tree may change a lot)
- Prediction performance is often poor (due to high variance)

Ensemble methods

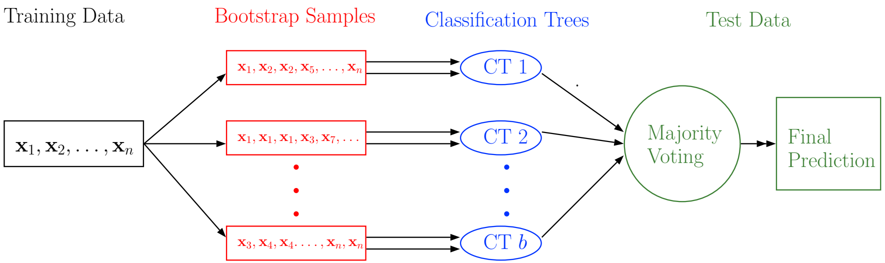
Main idea: build many classification trees independently and then vote the predictions of individual trees.



Two options: **Bagging** (Bootstrap AGGregatING) and **Random Forest**.

Bagging

In bagging each classification tree is built on a separate **bootstrap sample** of training data (i.e., a random sample with replacement).

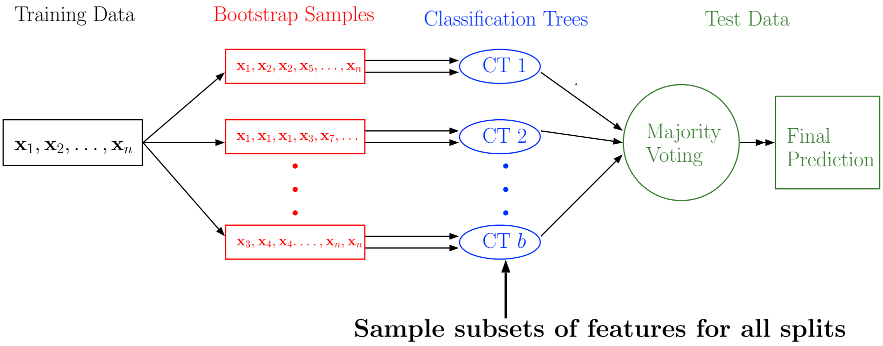

Training Data
Bootstrap Samples
Classification Trees
Test Data
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_5, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_7, \ldots$
$\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_4 \ldots, \mathbf{x}_n, \mathbf{x}_n$
CT 1
CT 2
CT b
Majority Voting
Final Prediction

Some comments on bagging

- Averaging many trees decreases the variance of the model, without increasing the bias (as long as the trees are not correlated)
- Bootstrap sampling is a way of de-correlating the trees (as simply training many trees on a single training set would give strongly correlated trees)
- The number of bootstrap samples/trees, b, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set.

Random forest

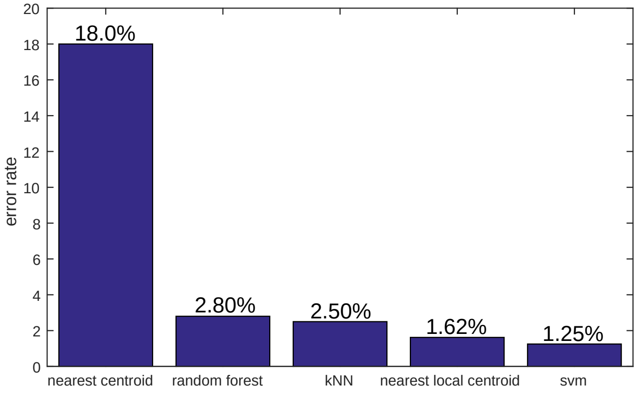
Random forest modifies bagging by **allowing every tree in the ensemble to randomly select predictors for decision splits**.


Training Data
Bootstrap Samples
Classification Trees
Test Data
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_5, \ldots, \mathbf{x}_n$
$\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_7, \ldots$
$\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_4. \ldots, \mathbf{x}_n, \mathbf{x}_n$
CT 1
CT 2
CT b
Majority Voting
Final Prediction
Sample subsets of features for all splits

Comments on random forest

- The motivation is to further de-correlate the trees in bagging: If one or a few features are very strong predictors for the class label, these features will be selected in many of the trees, causing them to become correlated.
- Typically, for a classification problem with d features, $\sqrt{d}$ features (selected at random) are used in each split.
- Random forest generally increases both speed and accuracy over bagging.

Methods comparison (including random forest)



Thank you for your attention

Summary of this presentation:

- Introduced classification + a dataset of handwritten digits
- Presented 3 major classifiers: kNN, SVM and random forest
- Not covered: discriminant analysis, logistic regression, neural nets

If you wish to learn more,

- visit: `http://www.math.sjsu.edu/~gchen/Math285S16.html`, or
- email: `guangliang.chen@sjsu.edu`