EE482/682 DSP APPLICATIONS

OBJECT RECOGNITION

OUTLINE

- Knowledge Representation
- Statistical Pattern Recognition
- Neural Networks
- Boosting
- Random Forests

NOTE

- These slides will follow parts of Sonka Chapter 9
  - Available through UNLV library (or VPN)

- Additional reading available through Szeliski
  - 1e Chapter 14: “Classical” recognition
  - 2e Chapter 6: Modern deep learning approaches
    - Will be covering these topics in more detail in the coming weeks

OBJECT RECOGNITION

- Pattern recognition is a fundamental component of machine vision
- Recognition is high-level image analysis
  - From the bottom-up perspective (pixels → objects)
  - Many software packages exist to easily implement recognition algorithms (e.g. Weka Project, R package)

- Goal of object recognition is to "learn" characteristics that help distinguish object of interest
  - Most are binary problems

KNOWLEDGE REPRESENTATION

- Syntax – specifies the symbols that may be used and ways they may be arranged
- Semantics – specifies how meaning is embodied in syntax
- Representation – set of syntactic and semantic conventions used to describe things

- Sonka book focuses on artificial intelligence (AI) representations
  - More closely related to human cognition modeling (e.g. how humans represent things)
  - Not as popular in computer vision community

DESCRIPTORS/FEATURES

- Most common representation in vision
- Descriptors (features) usually represent some scalar property of an object
  - These are often combined into feature vectors
- Numerical feature vectors are inputs for statistical pattern recognition techniques
  - Descriptor represents a point in feature space

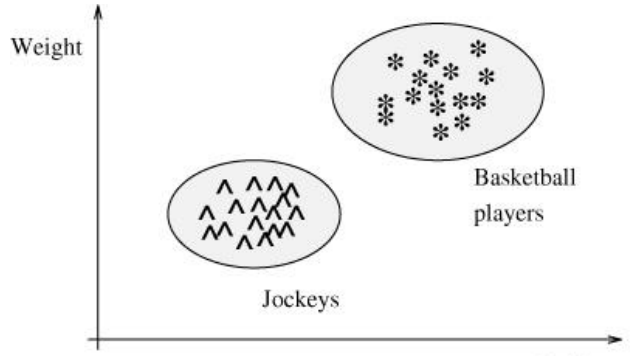


Figure 9.1: Recognition of basketball players and jockeys. © *Cengage Learning 2015.*

STATISTICAL PATTERN RECOGNITION

- Object recognition = pattern recognition
  - Pattern – measureable properties of object
- Pattern recognition steps:

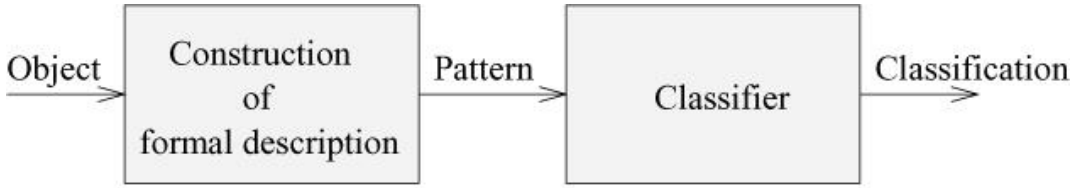
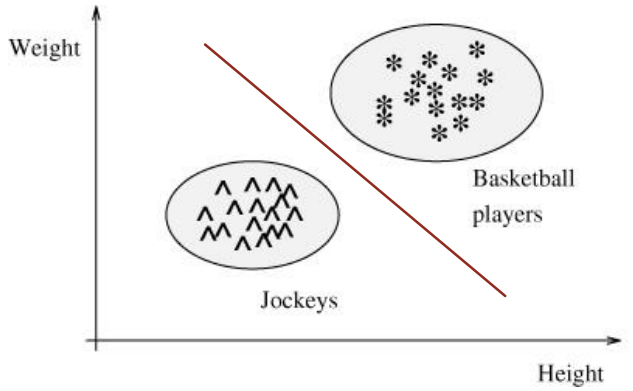


Figure 9.4: Main pattern recognition steps. © *Cengage Learning 2015.*

  - Description – determine right features for task
  - Classification – technique to separate different object "classes"
- Separable classes – hyper-surface exists perfectly distinguish objects
  - Hyper-planes used for linearly separable classes
  - This is unlikely in real-world scenarios

GENERAL CLASSIFICATION PRINCIPLES I

- A statistical classifiers takes in a n-dimensional feature of an object and has a single output
  - The output is one of the R available class symbols (identifiers)
- Decision rule – describes relations between classifier inputs and output
  - $d(\boldsymbol{x}) = \omega_r$
  - Divides feature space into R disjoint subsets K_r
- Discrimination hyper-surface is the border between subsets
- Discrimination function
  - $g_r(\boldsymbol{x}) \geq g_s(\boldsymbol{x}), s \neq r$
    - $\boldsymbol{x} \in K_r$
- Discrimination hyper-surface between class regions
  - $g_r(\boldsymbol{x}) - g_s(\boldsymbol{x}) = 0$



GENERAL CLASSIFICATION PRINCIPLES II

- Decision rule
  - $d(\boldsymbol{x}) = \omega_r \Leftrightarrow g_r(\boldsymbol{x}) = \max_{s=1,\ldots,R} g_s(\boldsymbol{x})$
  - Which subset (region) provides maximum discrimination
- Linear discriminant functions are simple and often used in linear classifier
  - $g_r(\boldsymbol{x}) = q_{r0} + q_{r1}x_1 + \cdots + q_{rn}x_n$
- Must use non-linear for more complex problems
  - Trick is to transform the original feature space into a higher dimensional space
    - Can use a linear classifier in the higher dimensional space
  - $g_r(\boldsymbol{x}) = \boldsymbol{q_r} \cdot \Phi(\boldsymbol{x})$
    - $\Phi(\boldsymbol{x})$ – non-linear mapping to higher-d space

NEAREST NEIGHBORS (NN) CLASSIFIER I

- Classifier based on minimum distance principle
- Minimum distance classifier labels pattern $\boldsymbol{x}$ into the class with closest exemplar
  - $d(\boldsymbol{x}) = argmin_s|\boldsymbol{v}_s - \boldsymbol{x}|$
  - $\boldsymbol{v}_s$ - exemplars (sample pattern) for class ω_s
- With a single exemplar per class, results in linear classifier

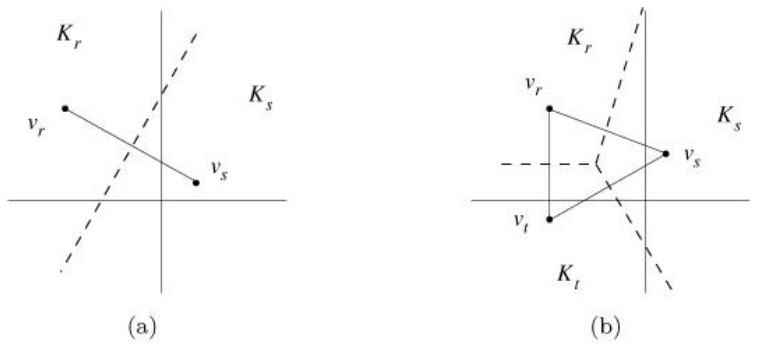


Figure 9.6: Minimum distance discrimination functions. © *Cengage Learning 2015.*

NEAREST NEIGHBORS (NN) CLASSIFIER II

- Very simple classifier uses multiple exemplars per class
  - Take same label as closest exemplar
- k-NN classifier
  - More robust version by examining k closest points and taking most often occurring label
- Advantage: easy "training"
- Problems: computational complexity
  - Scales with number of exemplars and dimensions
  - Must do many comparisons
  - Can improve performance with K-D trees

CLASSIFIER OPTIMIZATION I

- Discriminative classifiers are deterministic
  - Pattern $\boldsymbol{x}$ always mapped to same class
- Would like to have an optimal classifier
  - Classifier the minimizes the errors in classification
- Define loss function to optimize based on classifier parameters q
  - $J(q^*) = \min_q J(q)$
  - $d(x, q) = \omega$ – decision of classifier with params q given test example x
- Minimum error criterion (Bayes criterion, maximum likelihood) loss function
  - $\lambda(\omega_r|\omega_s)$ - loss incurred if classifier incorrectly labels object ω_r
    - $\lambda(\omega_r|\omega_s) = 1$ for $r \neq s$
- Mean loss
  - $J(q) = \int_X \sum_{s=1}^{R} \lambda(d(x,q)|\omega_s) p(x|\omega_s) p(\omega_s) dx$
    - $p(\omega_s)$ - prior probability of class
    - $p(x|\omega_s)$ - conditional probability density

CLASSIFIER OPTIMIZATION II

- Discriminative function
  - $g_{r(x)} = p(x|\omega_r)p(\omega_r)$
  - Corresponds to posteriori probability $p(\omega_r|x)$
- Posteriori probability describes how often pattern x is from class ω_r
- Optimal decision is to classify x to class ω_r if posteriori $p(\omega_r|x)$ is highest
  - However, we do not know the posteriori
- Bayes theorem
  - $p(\omega_s|x) = \frac{p(x|\omega_s)p(\omega_s)}{p(x)}$
- Since $p(x)$ is a constant and prior $p(\omega_s)$ is known,
  - Just need to maximize likelihood $p(x|\omega_s)$
- This is desirable because the likelihood is something we can learn using training data

CLASSIFIER TRAINING

- Supervised approach: Training set is given with feature and associated class label
  - $T = \{(\boldsymbol{x}_i, y_i)\}$
  - Used to set the classifier parameters $\boldsymbol{q}$
- Learning methods should be inductive to generalize well
  - Represent entire feature space
  - E.g. work even on unseen examples

CLASSIFIER TRAINING II

- Usually, larger datasets result in better generalization
  - Some state-of-the-art classifiers use millions of examples
  - Try to have enough samples to statistical cover space
- N Cross-fold validation/testing
  - Divide training data into a train and validation set
  - Only train using training data and check results on validation set
  - Can be used for "bootstrapping" or to select best parameters after partitioning data N times

CLASSIFIER LEARNING

- Probability density estimation
  - Estimate the probability densities $p(\boldsymbol{x}|\omega_r)$ and priors $p(\omega_r)$
- Parametric learning
  - Typically, the distribution $p(\boldsymbol{x}|\omega_r)$ shape is known but the parameters must be learned
    - E.g. Gaussian mixture model
  - Like to select a distribution family that can be efficiently estimated such as Gaussians
  - Prior estimation by relative frequency
    - $p(\omega_r) = K_r/K$
      - Number of objects in class r over total objects in training database

SUPPORT VECTOR MACHINES (SVM)

- Maybe the most popular classical classifier
  - Good generalizability even with limited data
- SVM is an optimal classification for separable two-class problem
  - Maximizes the margin (separation) between two classes → generalizable and avoids overfitting
  - Relaxed constraints for non-separable classes
  - Can use kernel trick to provide non-linear separating hyper-surfaces
- Support vectors – vectors from each class that are closest to the discriminating surface → define the margin
- Rather than explicitly model the likelihood, search for the discrimination function
  - Don't waste time modeling densities when class label is all we need

SVM INSIGHT

- SVM is designed for binary classification of linearly separable classes
- Input $\boldsymbol{x}$ is n-dimensional (scaled between [0,1] to normalize) and class label $\omega \in \{-1,1\}$
- Discrimination between classes defined by hyperplane such that no training samples are misclassified
  - $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
    - $\boldsymbol{w}$ – plane normal, b offset
  - Optimization finds "best" separating hyperplane

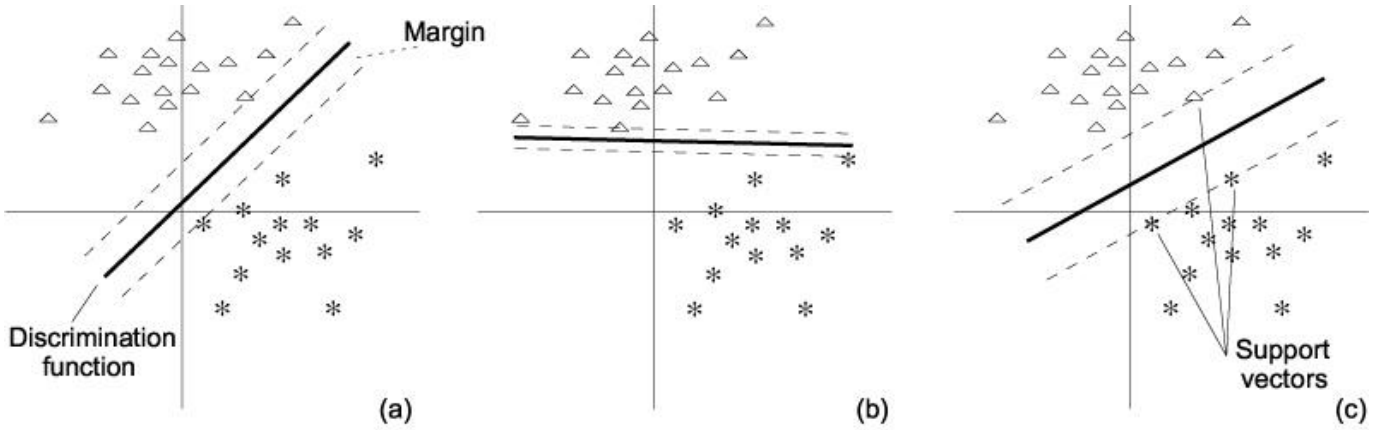


Figure 9.9: Basic two-class classification idea of support vector machines. (a) and (b) show two examples of non-optimal linear discrimination. (c) An optimal linear discriminator maximizes the margin between patterns of the two classes. The optimal hyperplane is a function of the support vectors. © *Cengage Learning 2015.*

SVM POWER

- Final discrimination function
  - $f(x) = w \cdot x + b$
- Re-written using training data
  - $f(x) = \sum_{i \in SV} \alpha_i \omega_i (x_i \cdot x) + b$
    - α_i - weight of support vector SV
  - Only need to keep support vectors for classification
- Kernel trick – replace $(x_i \cdot x)$ with non-linear mapping kernel
  - $k(x_i, x) = \Phi(x_i) \cdot \Phi(x_j)$
  - For specific kernels this can be efficiently computed without doing the warping Φ
    - Can even map into an infinite dimensional space
  - Allows linear separation in a higher dimensional space

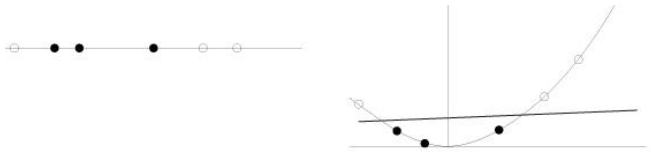


Figure 9.10: Achieving linear separability by application of a kernel function. On the left, the two classes are not linearly separable in 1D: on the right, the function $\Phi(x) = x^2$ creates a linearly separable problem. © *Cengage Learning 2015.*

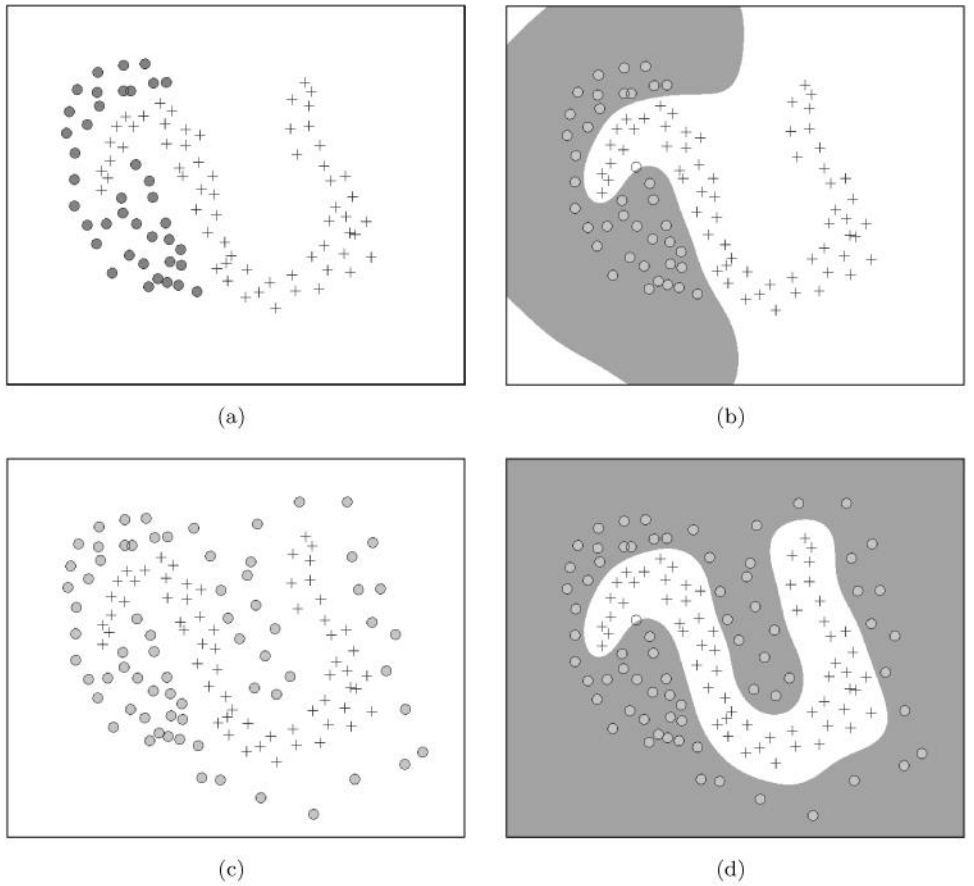


Figure 9.11: Support vector machine training; Gaussian radial basis function kernel used (equation 9.49). (a,c) Two-class pattern distribution in a feature space. (Note that the "+" patterns in (a) and (b) are identical while the "o" patterns in (a) are a subset of patterns in (b)). (b,d) Non-linear discrimination functions obtained after support vector machine training. © *Cengage Learning 2015.*

SVM RESOURCES

- More detailed treatment can be found in
  - Duda, Hart, Stork, "Pattern Classification"
- Lecture notes from Nuno Vasconcelos (UCSD)
  - http://www.svcl.ucsd.edu/courses/ece271B-F09/handouts/SVMs.pdf

- SVM software
  - LibSVM (Java) [link]
  - SVMLight (C) [link]
  - Scikit-learn (Python) [link]

CLUSTER ANALYSIS

- Unsupervised learning method that does not require labeled training data
- Divide training set into subsets (clusters) based on mutual similarity of subset elements
  - Similar objects are in a single cluster, dissimilar objects in separate clusters
- Clustering can be performed hierarchically or non-hierarchically
- Hierarchical clustering
  - Agglomerative – each sample starts as its own cluster and clusters are merged
  - Divisive – the whole dataset starts as a single cluster and is divided
- Non-hierarchical clustering
  - Parametric approaches – assumes a known class-conditioned distribution (similar to classifier learning)
  - Non-parametric approaches – avoid strict definition of distribution

K-MEANS CLUSTERING

- Very popular non-parametric clustering technique
  - Based on minimizing the sum of squared distances
    - $E = \sum_{i=1}^{K} \sum_{x_j \in V_i} d^2(x_j, v_i)$
  - Simple and effective
- K-means algorithm
  - Input is n-dimensional data points and number of clusters K
  - Initialize cluster starting points
    - $\{v_1, v_2, \ldots, v_K\}$
  - Assign points to closest v_i using distance metric d
  - Recompute v_i as centroid of associated data V_i
  - Repeat until convergence

K-MEANS DEMO

- https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

- http://alekseynp.com/viz/k-means.html

NEURAL NETWORKS

- Early success on difficult problems
  - Renewed interest with deep learning
- Motivated by human brain and neurons
  - Neuron is elementary processor which takes a number of inputs and generates a single output
- Each input has associated weight and output is a weighted sum of inputs

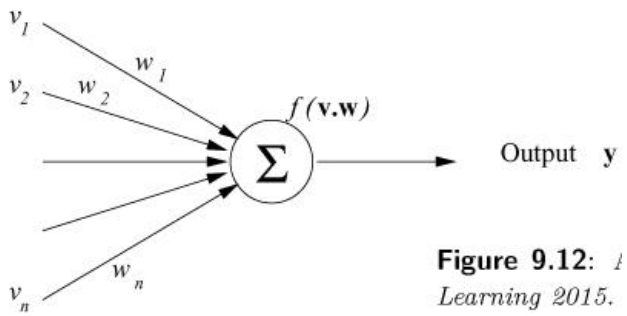


Figure 9.12: *Learning 2015.*

- The network is formed by interconnecting neurons
  - Outputs of neurons as inputs to others
  - May have many inputs and many outputs

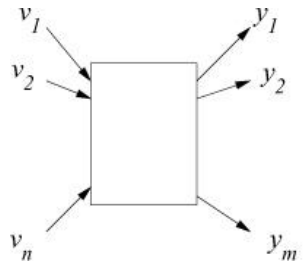


Figure 9.13: *gage Learning*

- NN tasks:
  - Classification – binary output
  - Auto-association – re-generate input to learn network representation
  - General association – associations between patterns in different domains

NN VARIANTS

- Feed-forward networks

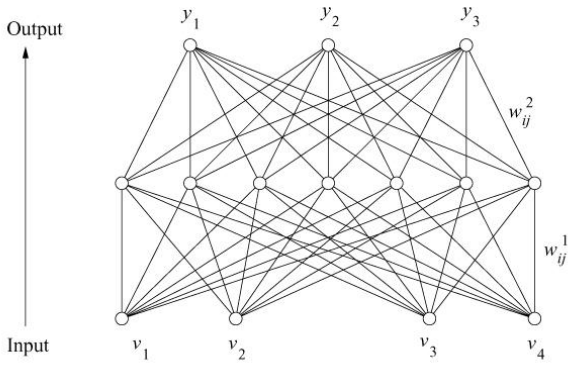


Figure 9.14: A three-layered neural net structure. © *Cengage Learning 2015.*

  - Include "hidden" layers between input and output
- Can handle more complicated problems
- Networks "taught" using back-propagation
  - Compare network output to expected (truth) output
  - Minimize SSD error by adjusting neuron weight

- Kohonen feature maps
  - Unsupervised learning that organizes network to recognize patterns
- Performs clustering
  - Neighborhood neurons are related

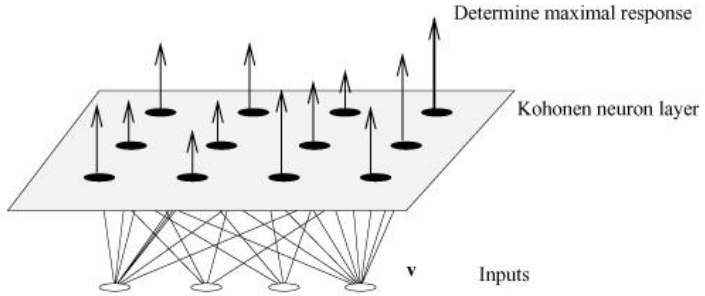


Figure 9.15: Kohonen self-organizing neural net. © *Cengage Learning 2015.*

- Network lies on a 2D layer
  - Fully connect neurons to all inputs
  - Neuron with highest input
    - $x = \sum_{i=1}^{n} v_i w_i$
  - is the winner (cluster label)

BOOSTING

- Generally, a single classifier does not solve problem well enough
  - Is it possible to improve performance by using more classifiers (e.g. experts)?
- Boosting – intelligent combination of weak classifiers to generate a strong classifier
  - Weak classifier works a little better than chance (50% for binary problem)
  - Final decision rule combines each weak classifier output by weighted confidence majority vote
    - $C(x) = sign(\sum_i \alpha_i C_i(x))$
      - α_i - confidence in classifier $C_i(.)$
- Training
  - Sequentially train classifiers to focus classification effort on "hard" examples
  - After each training round, re-weight misclassified examples

- Advantages:
  - Generally, does not overfit but is able to achieve high accuracy
    - Training rounds increase margin
  - Many modification exist to improve performance
    - Gentle and BrownBoost for outlier robustness
    - Strong theoretical background
  - Flexible with only "weak" classifier requirement
    - Can use any type of classifier (statistical, rule-based, of different type, etc.)

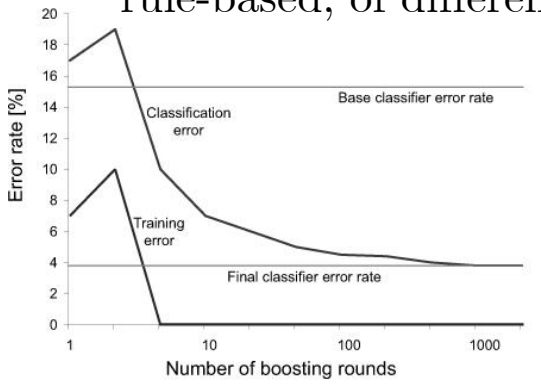


Figure 9.34: AdaBoost: training and testing error rate curves plotted against the number of boosting rounds. Note that testing error keeps decreasing long after training error has already reached zero. This is associated with a continuing increase in the margin that increases the overall classification confidence with additional rounds of boosting. © *Cengage Learning 2015.*

RANDOM FORESTS I

- Classifier well suited for problems with many classes and large training datasets
  - Handle multiple classes, probabilistic output, generalizable, etc.
- Extension of decision tree to multiple trees in random fashion
- Two tasks
  - Classification – output nodes are class labels
  - Regression – output node gives continuous numeric value

RANDOM FORESTS II

- Decision tree structure (classification)
  - Simple sequential decision making
- Input at the tree root (top node)
- Split node – divide dataset based on one input feature
- Leaf node – output final class label
- Similar to boosting weak predictors for strong classifier

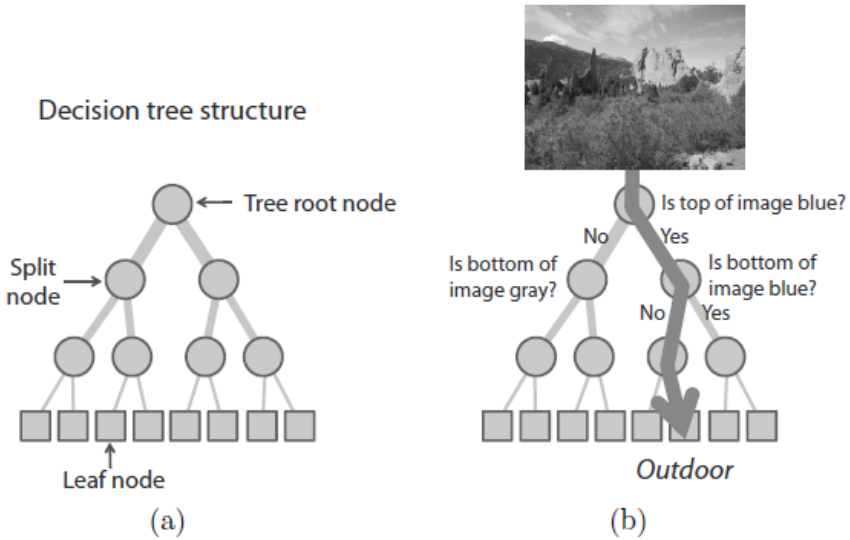
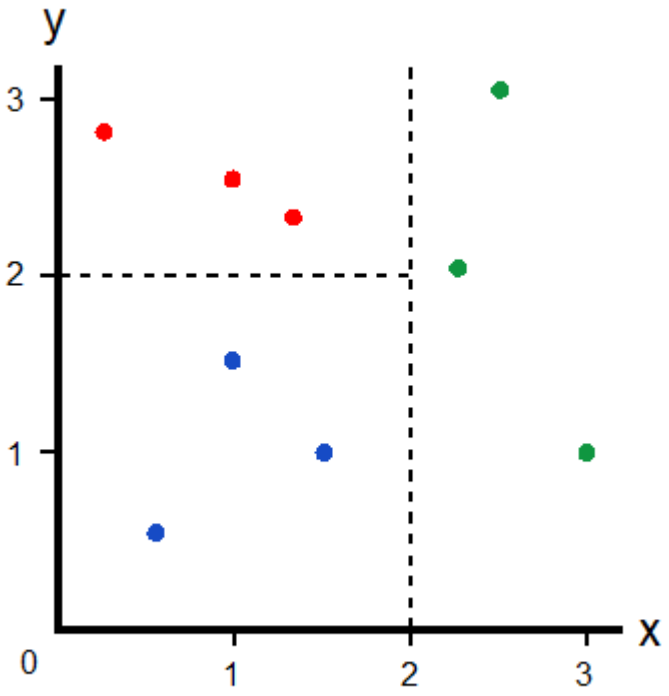
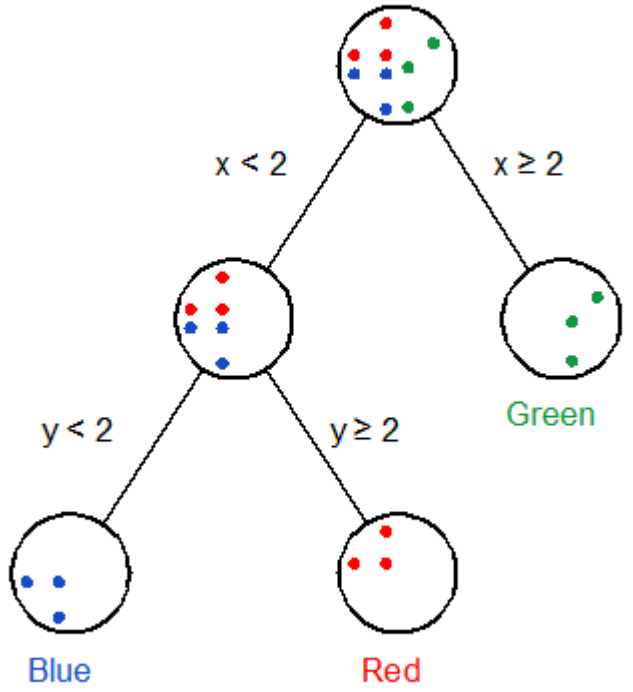


Figure 9.35: Decision tree. (a) Decision trees contain one root node, internal or split nodes (circles), and terminal or leaf nodes (squares). (b) A pattern arrives at the root and is sequentially passed to one of two children of each split node according to the node-based split function until it reaches a leaf node. Each leaf node is associated with a probability of a specific decision, for example associating a pattern with a class label. *Based on [Criminisi et al., 2011]. A color version*

RANDOM FORESTS III

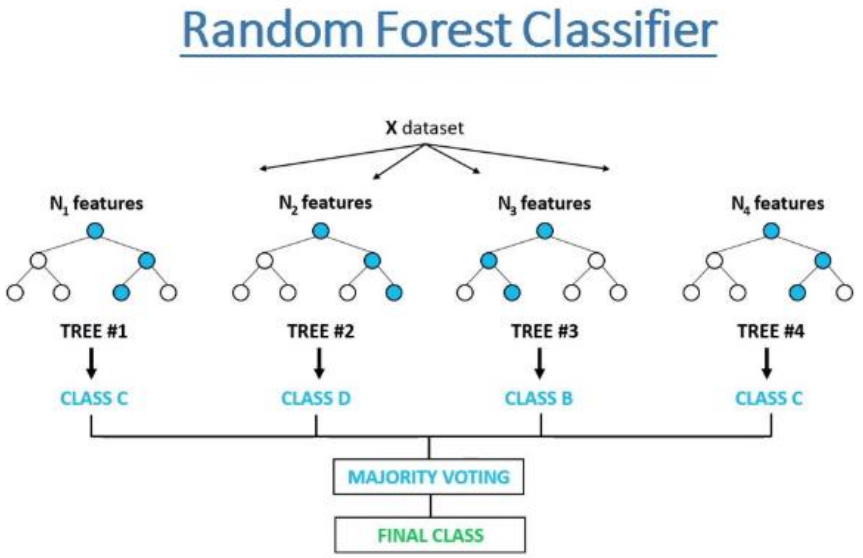




https://victorzhou.com/blog/intro-to-random-forests/

RANDOM FOREST IV

- Collect multiple decision trees and rely on majority vote for output class (ensemble model)
- Relies on randomness to have diverse (uncorrelated) trees
  - Bagging (bootstrap aggregation) – random sampling of training data with replacement
    - May use a training sample multiple times
    - DTs are sensitive to training data
  - Feature randomness – only use a subset of features at every split node



MODERN RECOGNITION

- Image classification – given an image, describe what is present
  - Single dominant class or multiple classes
- Object detection – place bounding box around every object class in image
  - Identification and localization
- Semantic segmentation – class label for every pixel in image
  - Dense classification
  - Panoptic segmentation for class and instance segmentation

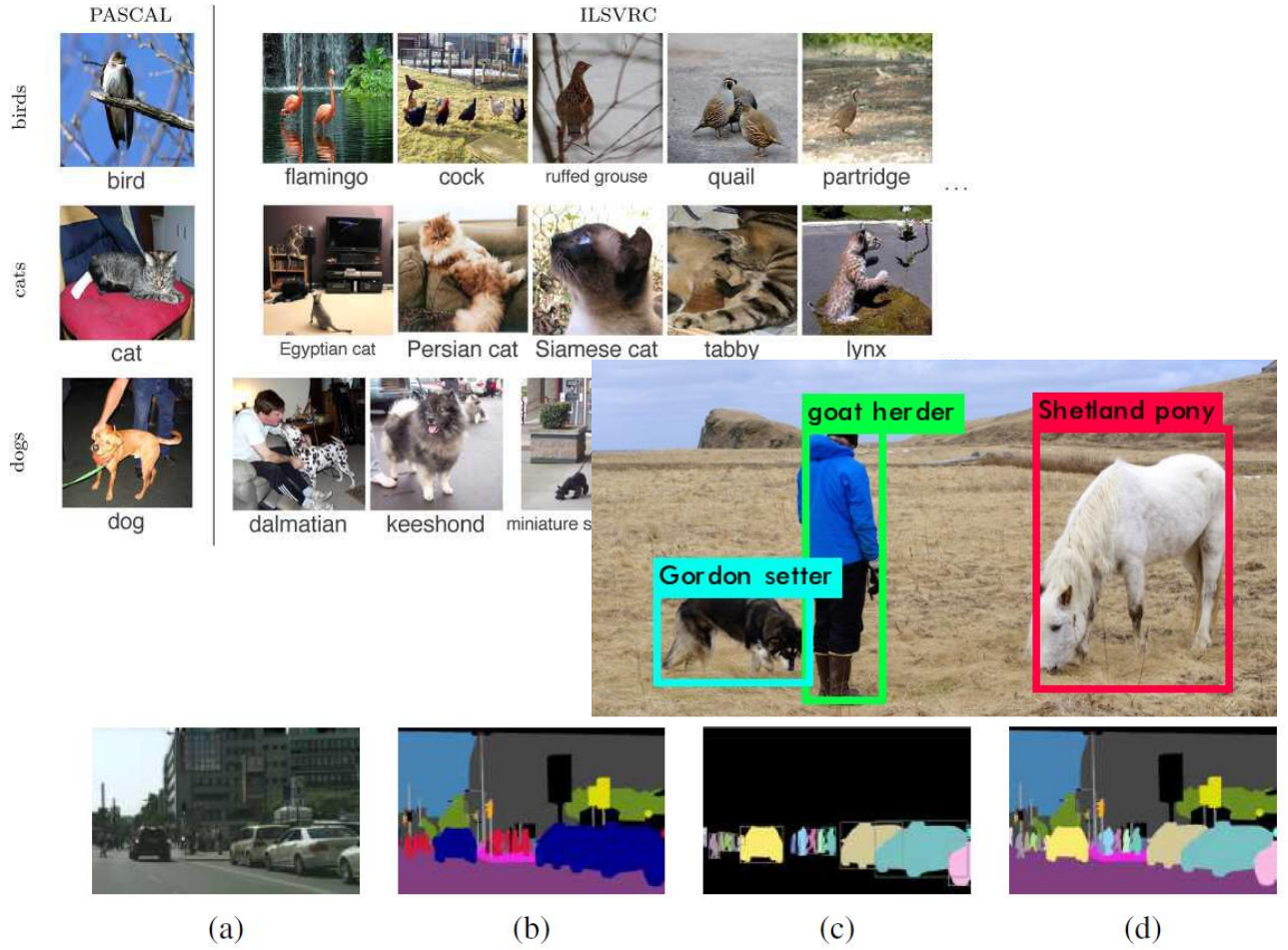


Figure 6.32 *Examples of image segmentation (Kirillov, He* et al. *2019) © 2019 IEEE: (a) original image; (b) semantic segmentation (per-pixel classification); (c) instance segmentation (delineate each object); (d) panoptic segmentation (label all things and stuff).*