Introduction to Machine Learning and Deep Learning

Jenq-Neng Hwang, Professor
Department of Electrical & Computer Engineering
University of Washington, Seattle WA
hwang@uw.edu

Grading Policy and Objectives

- **Grading Policy:** 5 biweekly Google Colab deep learning projects (20% each), and related paper reading reports
- **Learning Objectives:** Providing students with the fundamental skills and hands-on experience in applying the deep learning theories learned to various image/video/radar/lidar processing applications based on cloud based CPU/GPU computing resources. The course also involves biweekly projects and reading reports on provided real-world training and testing data, and paper reading assignments, for students' deep learning designs and report writing.

Paper Reading Report

- **On your own words (25% of each Hw), 3 pages max**
  - Motivation – Why do the authors want to work on this problem?
  - Contributions – What are the accomplishments they achieved in this paper (others did not achieve)?
  - Formulations – How do they solve the problems as mentioned/discussed in the introduction or related literatures?
  - Justification – Do the experiments/simulations support their claimed accomplishments?
  - Your Own Thoughts – what are you most impressed on this paper?

COVID-Related Policy

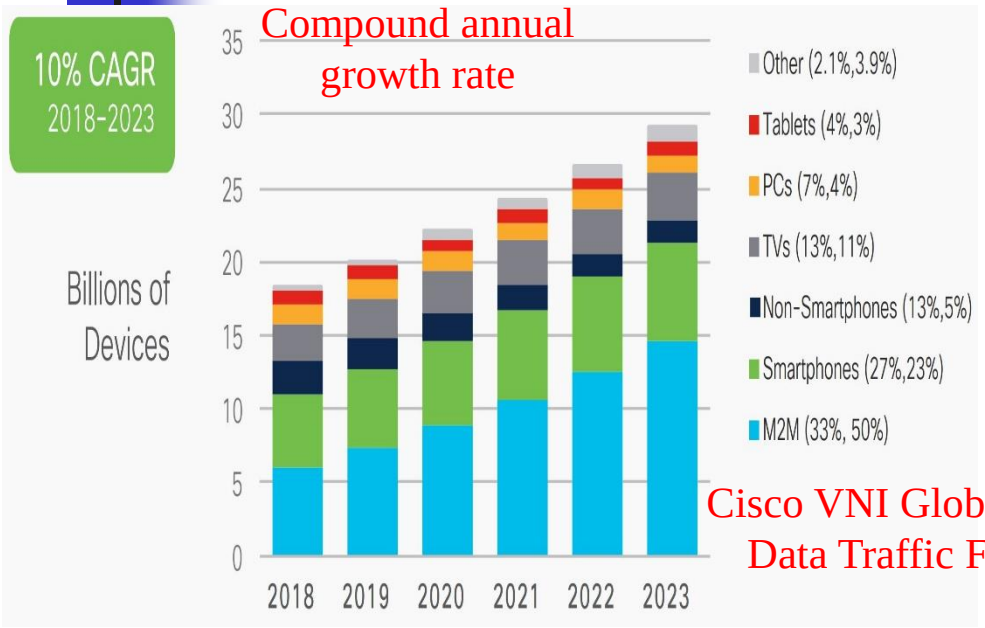
- **Please Wear a Mask (It's Mandatory.)**
  - Everyone—instructors and students alike—is required to wear a mask while indoors. As a result, I will be wearing a mask while lecturing, and you will be required to wear a mask, also. Eating and drinking are therefore prohibited in the classroom, although quick sips of water are permitted. Any student who refuses to wear a mask will be asked to leave the classroom. Students who refuse to comply with these requests and policies will be referred to the Community Standards and Student Conduct Office for possible disciplinary action.
  - In general, I expect you to comply with the UW Covid Protocols. https://www.washington.edu/coronavirus/autumn-quarter-health-and-safety/

- **If You're ill, Please Stay Home!**
  - Students who are ill are always advised to stay home, both for their own health, and for the health and safety of others. Never has this been more important than in the current COVID conditions. Please **do not** come to class if you are feeling unwell or have any cold or flu symptoms, and course, if you have COVID-19 symptoms. Stay home and take care of yourself! Email or message me to let me know you will be missing class due to illness and I will make arrangements for you to make up the class work or assignment, whenever possible.

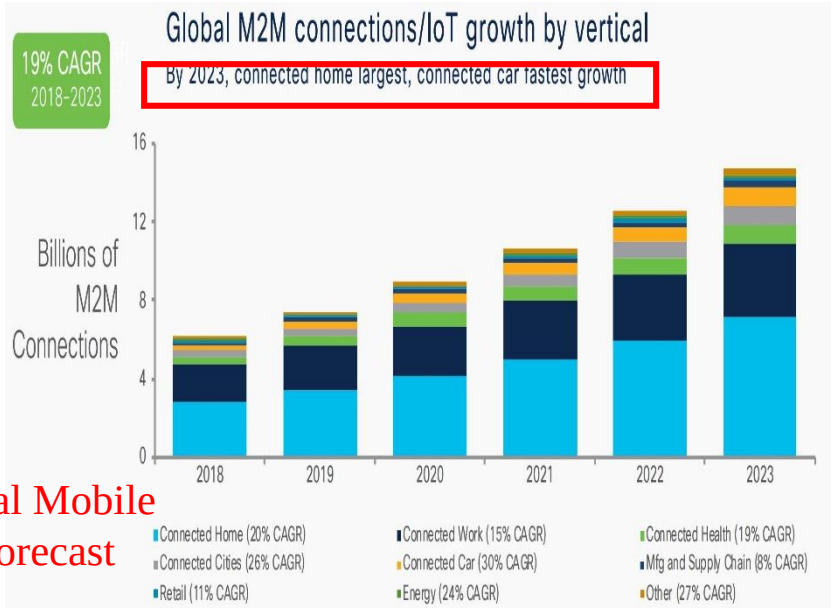
UW Policy on Religious Accommodations

- Washington state law requires that UW develop a policy for accommodation of student absences or significant hardship due to reasons of faith or conscience, or for organized religious activities.
- The UW's policy, including more information about how to request an accommodation, is available at Religious Accommodations Policy (https://registrar.washington.edu/staffandfaculty/religious-accommodations-policy/).
- Accommodations must be requested within the first two weeks of this course using the Religious Accommodations Request form (https://registrar.washington.edu/students/religious-accommodations-request/).

Exponential Growth of IoTs



Cisco VNI Global Mobile Data Traffic Forecast

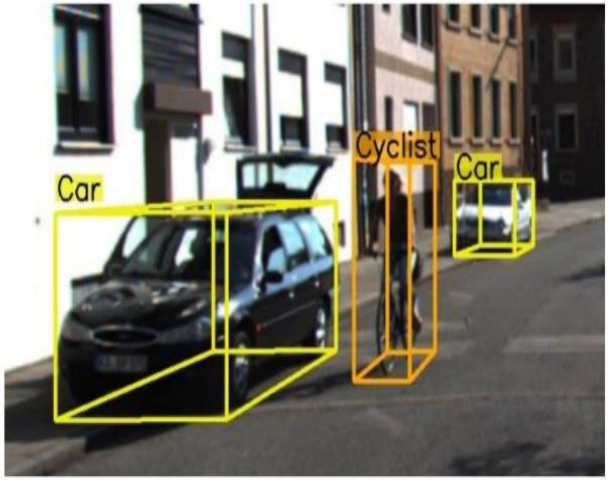


- M2M/IoT connections will grow 2.4-fold, from 6.1 billion in 2018 to 14.7 billion by 2023 → big data
- Increase of deployment of video applications on M2M/IoT, such as telemedicine and smart car navigation systems, which require greater bandwidth and lower latency

Uplinking Camera Videos

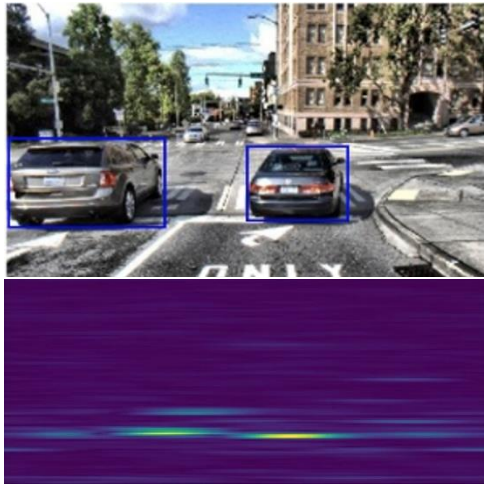
Various IoT Visual Sensors

Camera



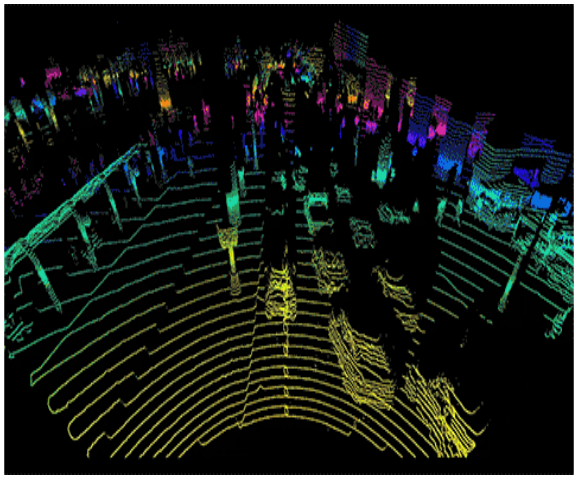
- Rich RGB information
- Sensitive to lighting condition
- Lack of depth Information

Radar



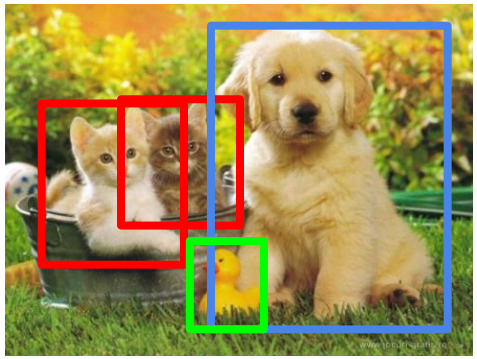
- Accurate Position/Velocity
- Can operate in bad weather or nights
- Poor in detection of small objects, ambiguity of objects types

Lidar



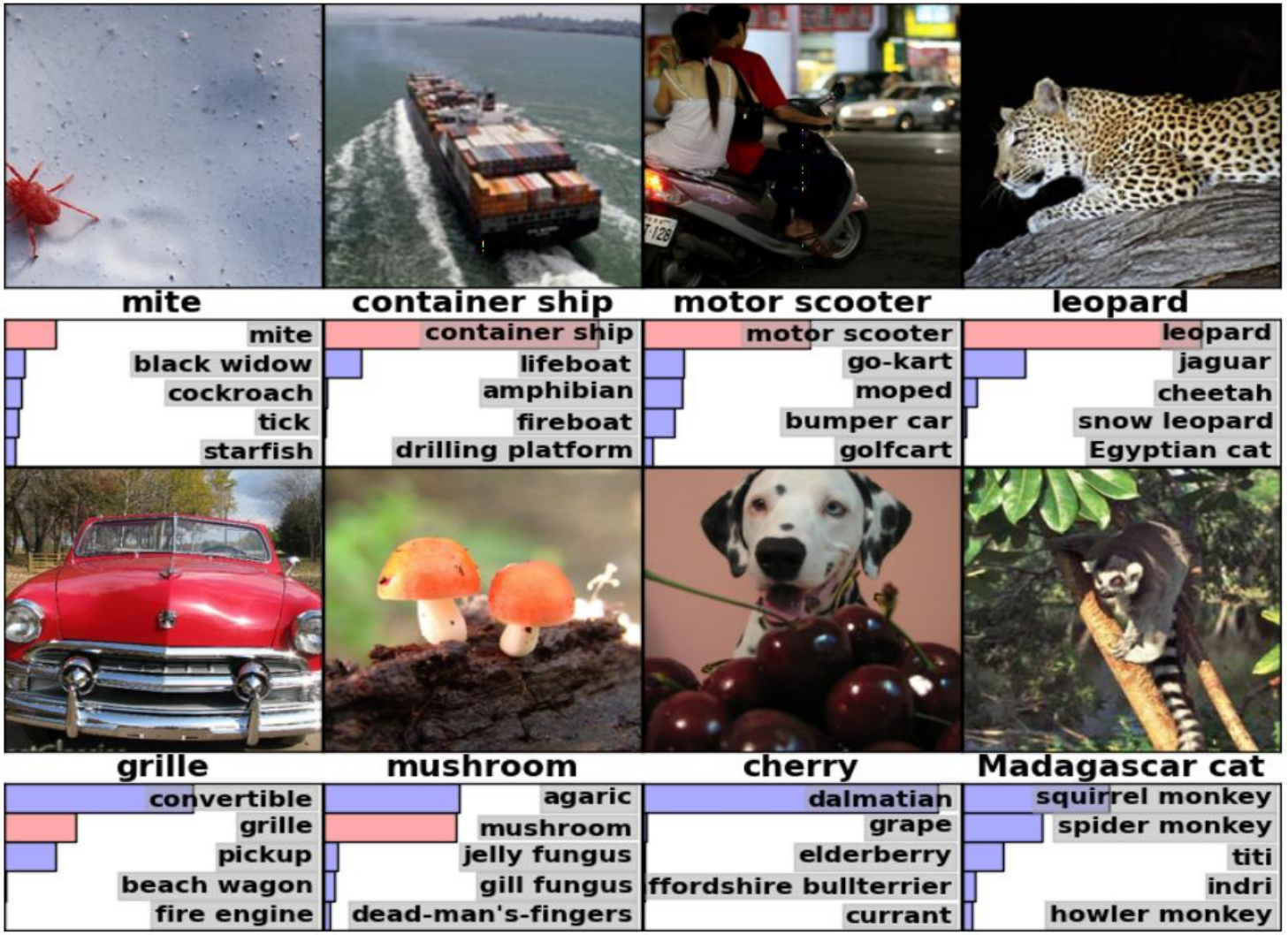
- High Precision;
- High cost and computation power
- Poor in small or far away objects (not enough cloud points)

Visual Machine Learning
(Image Based)

Classification	Classification + Localization	Object Detection	Instance Segmentation
			
CAT	CAT	CAT, DOG, DUCK	CAT, DOG, DUCK
Single object		Multiple objects	

Fei-Fei Li et al., "CS 231 lecture Note," Stanford Univ. 2018

ImageNet Data Classification

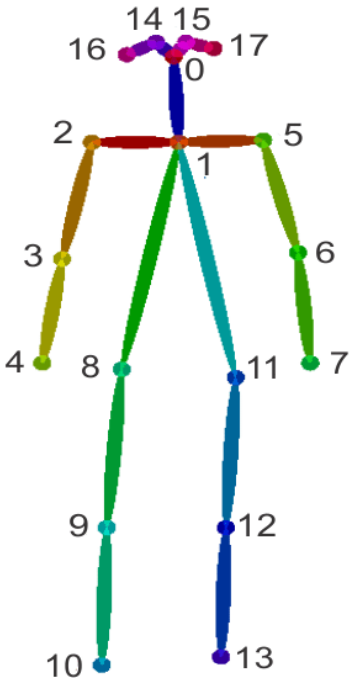


Alex Krizhevsky, et al. "ImageNet Classification with Deep Convolutional Neural Networks," NIPS 2012.

Image based Detection + Instance Segmentation

K. He, et al., “Mask R-CNN,” IEEE CVPR 2017

Image to 2D Pose (Skeleton)

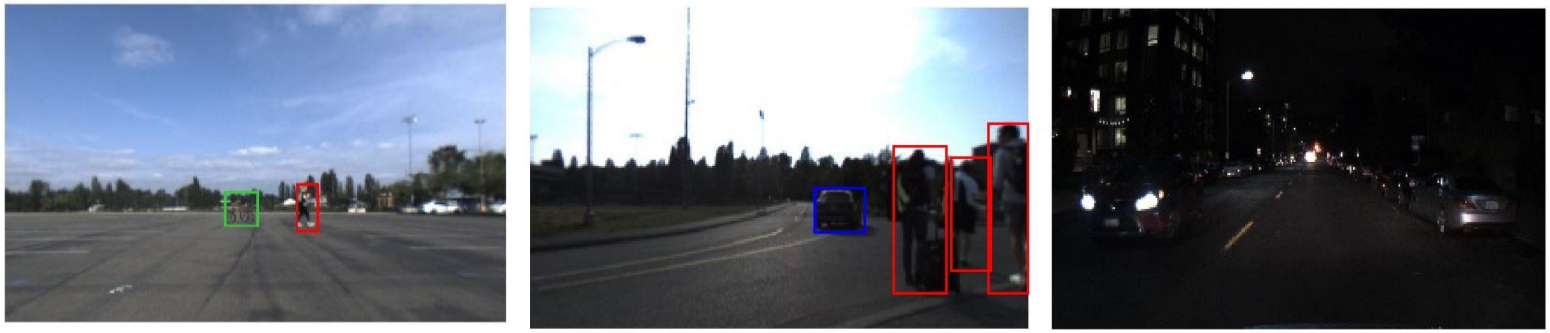


2D body keypoints

Z. Cao, T. Simon, S. E. Wei and Y. Sheikh, "Real-time multi-person 2d pose estimation using part affinity fields," *IEEE* CVPR, 2017.

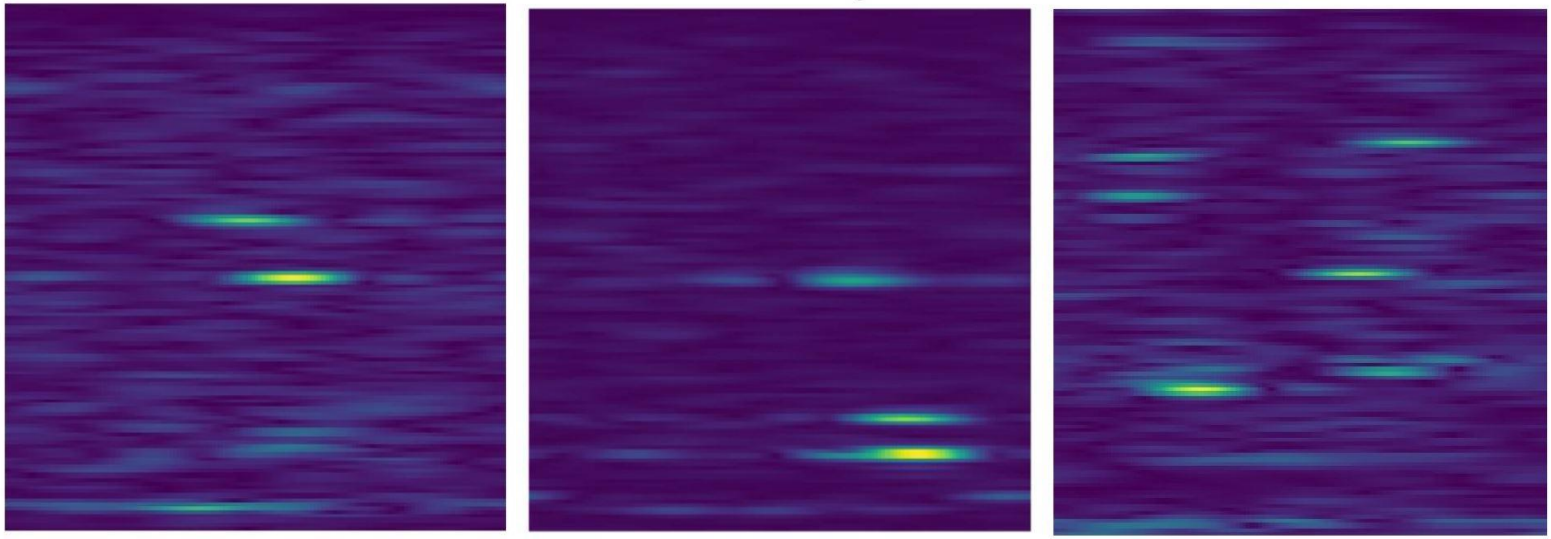
Radar Object Detection

camera



Radar RF images

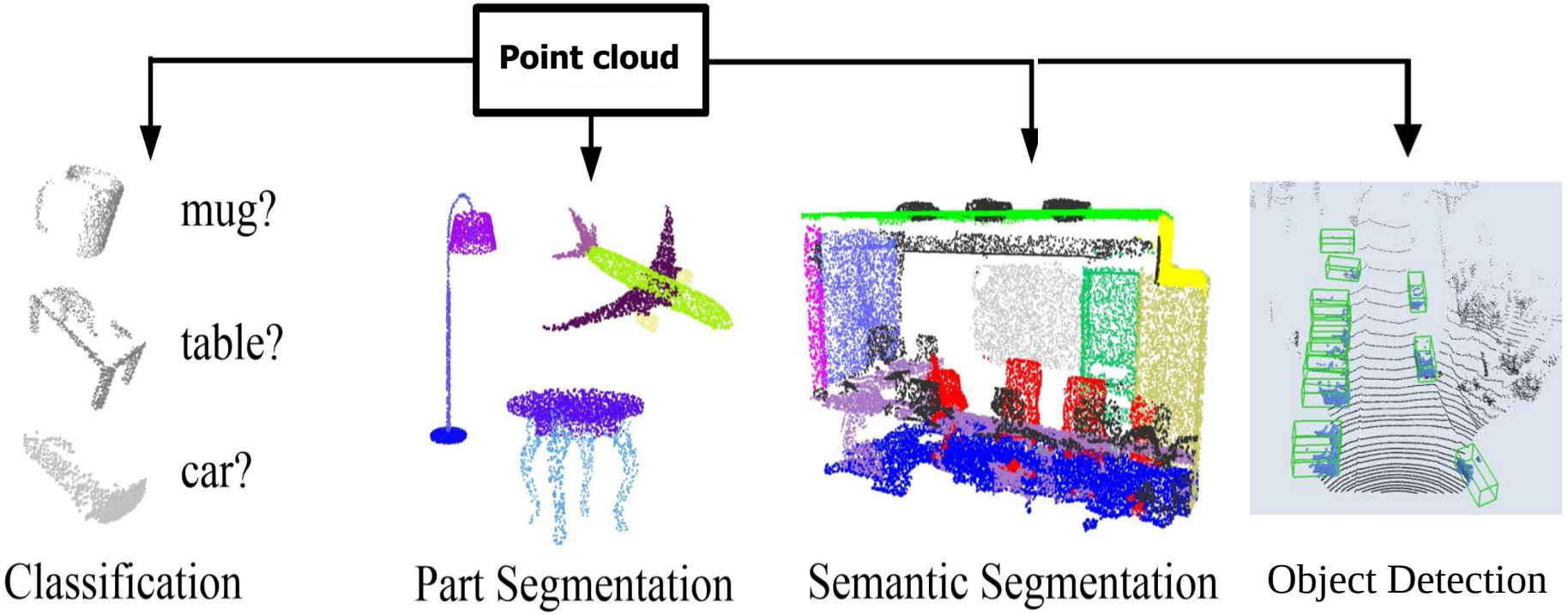
range



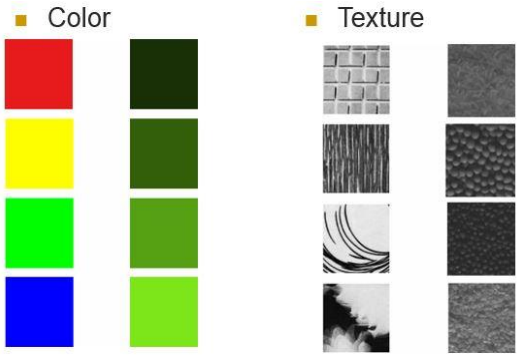
azimuth (angle)

Visual Tasks of Point Cloud

- Close to raw sensor data (with rich geometric information).
- Unordered point set as the input, partial views

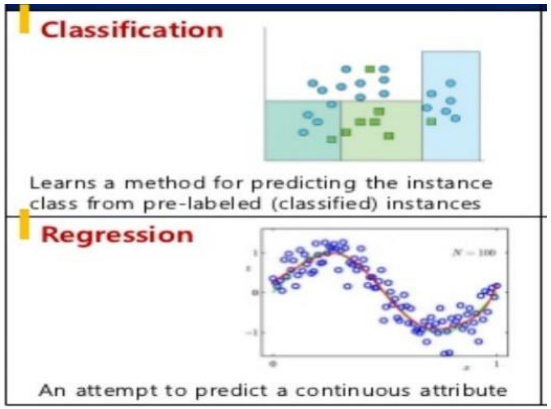


Two-Stage Visual Data Analytics



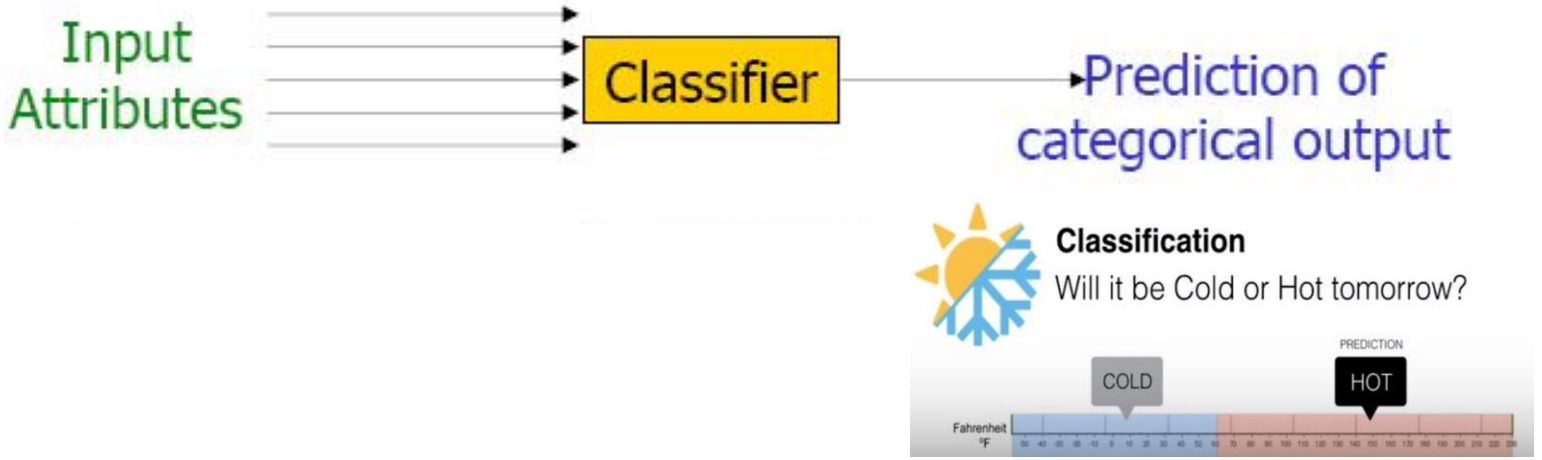
Supervised, Unsupervised, reinforcement learning, active learning, etc

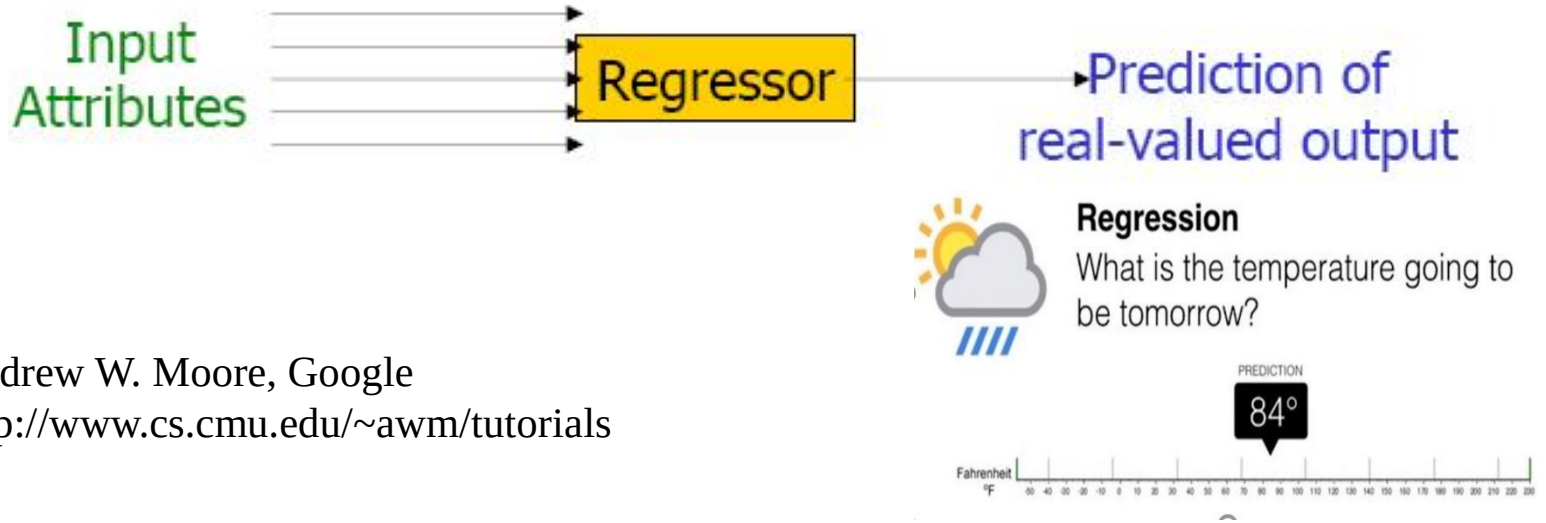
Visual Sensor → Feature Representation → Learning Algorithms → Applications



classification or regression, prediction, detection, etc

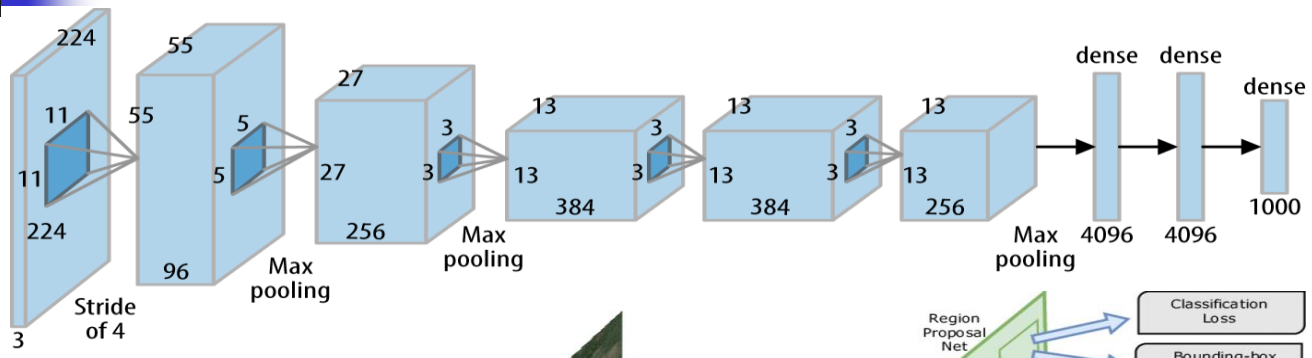
Two Common Types of Applications



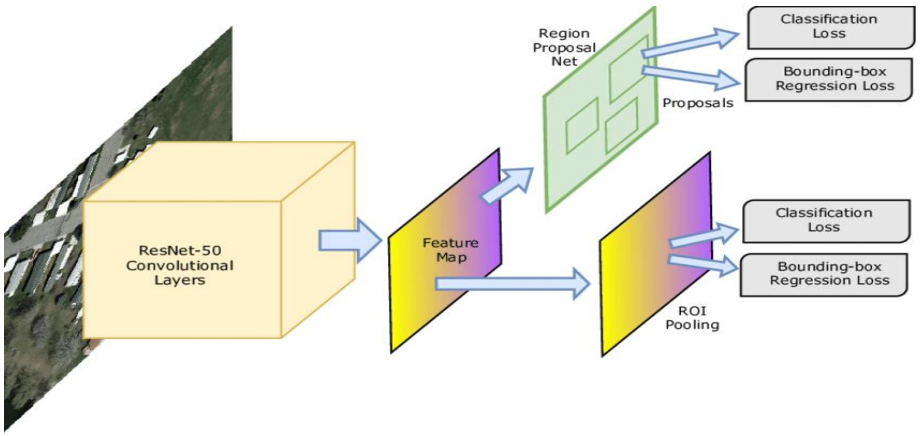


Andrew W. Moore, Google
http://www.cs.cmu.edu/~awm/tutorials

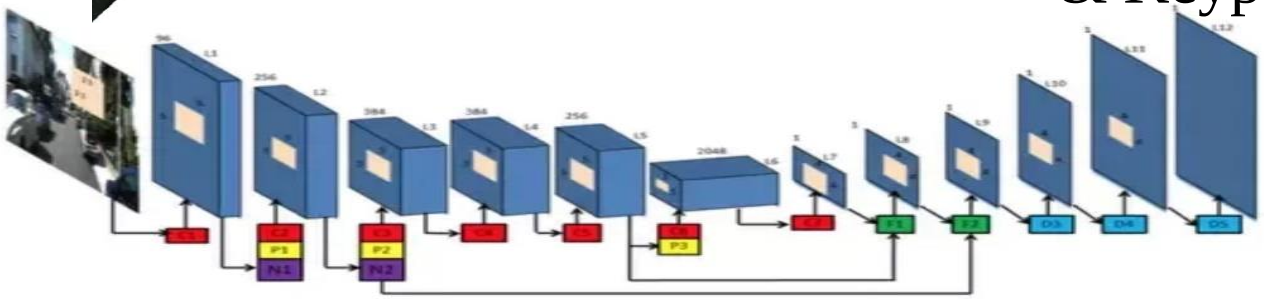
Deep Learning Architectures for End-to-End Analysis



Classification



Detection

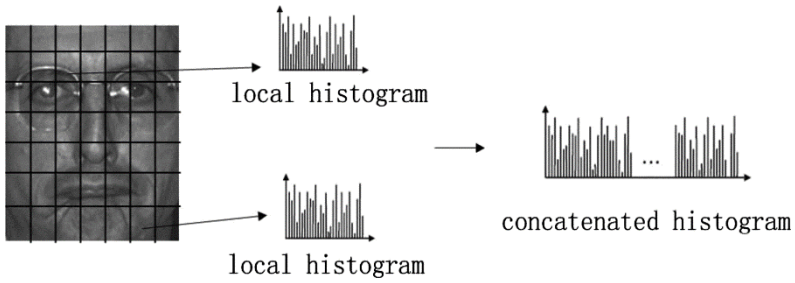


Segmentation & Keypoint

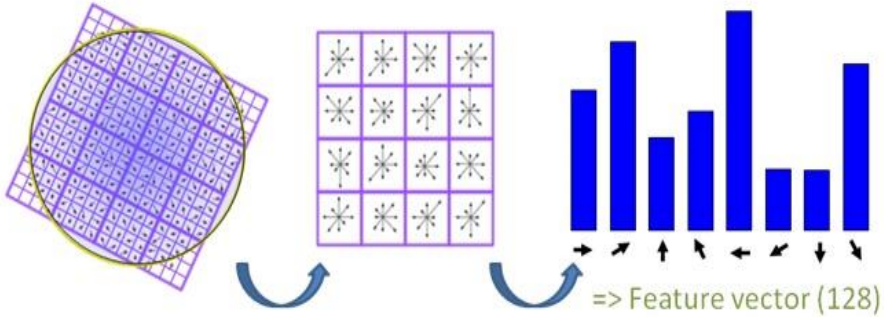
Course Outlines (10 weeks)

1. Introduction to Machine Learning and Deep Learning
2. Backpropagation, Multilayer Perceptron and Convolution Neural Networks, Pytorch and Google CoLab (HW1)
3. Convolution Neural Networks for Classification and Regression.
4. Deep Learning for Generative Adversarial Network (GAN) (HW2)
5. Few Shot Learning and Open Long-Tailed Recognition and Domain Adaptation
6. Deep Learning for Detection and Segmentation (HW3)
7. Deep Learning for Image/Video Related Applications
8. Deep Learning for Radar/Lidar Related Applications (HW4)
9. Recurrent Neural Networks and Graph Neural Networks
10. Self-Attention and Transformer for Seq2Seq Applications (HW5)

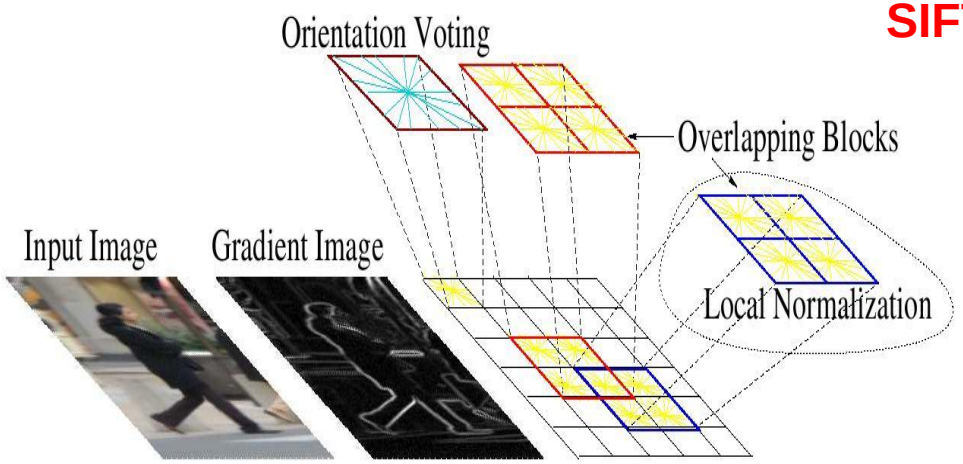
Common Visual Features



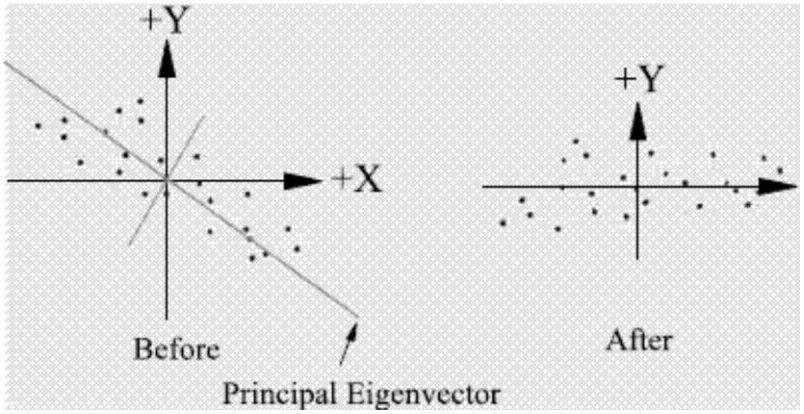
LBP (texture), face recognition



SIFT (interest points), object classification

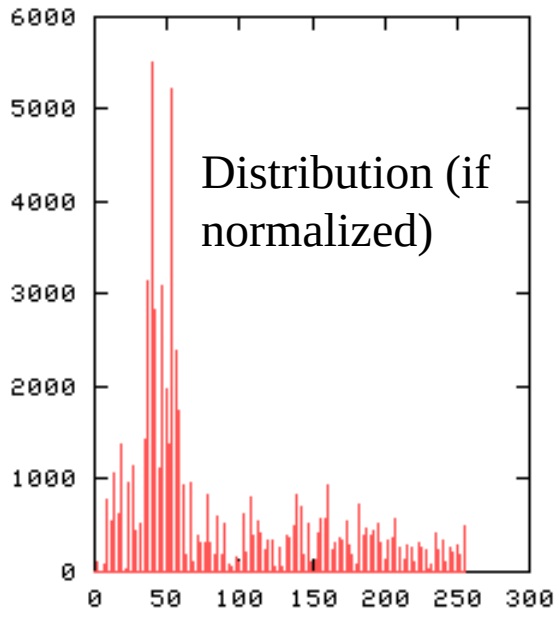


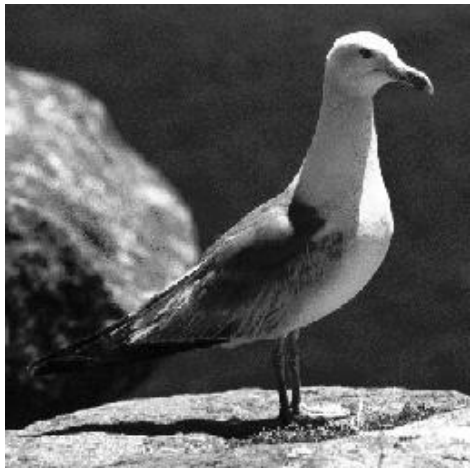
HOG (gradient), human detection

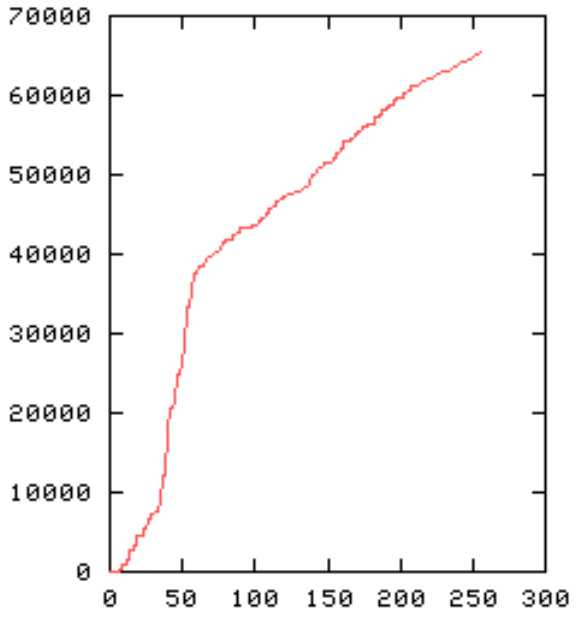


KL coefficients (PCA), face recognition

Image Representations: Histograms (Distributions)



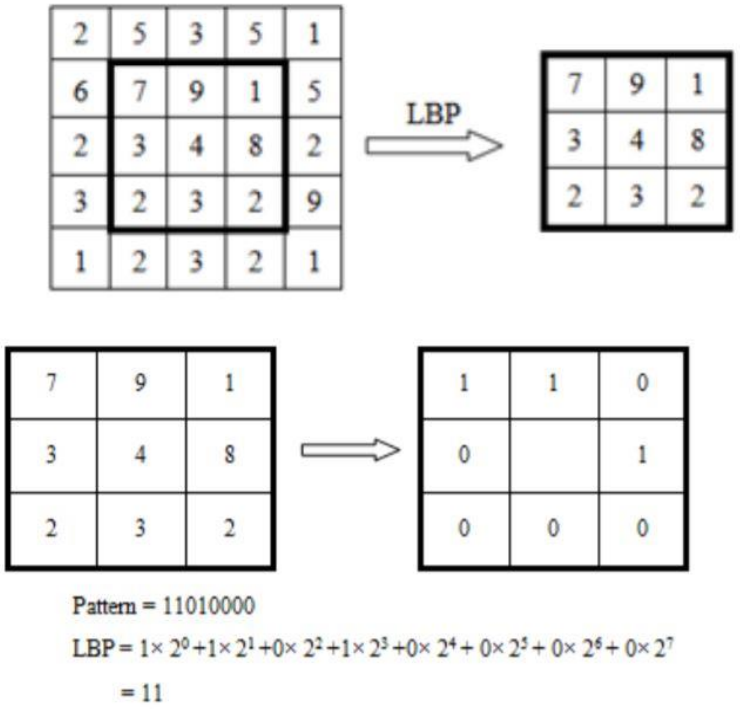




- Histogram and Cumulative Histogram (positive values)
- Can be generalize to higher dimensions (e.g., RGB channels)
- Different distribution of features (e.g., color, texture, depth, …)

Local Binary Pattern (LBP)

Any pixel *p*, create an 8-bit number *b1 b2 b3 b4 b5 b6 b7 b8*, where *bi = 0* if neighbor *i* has value less than *p's* value and 1 otherwise.



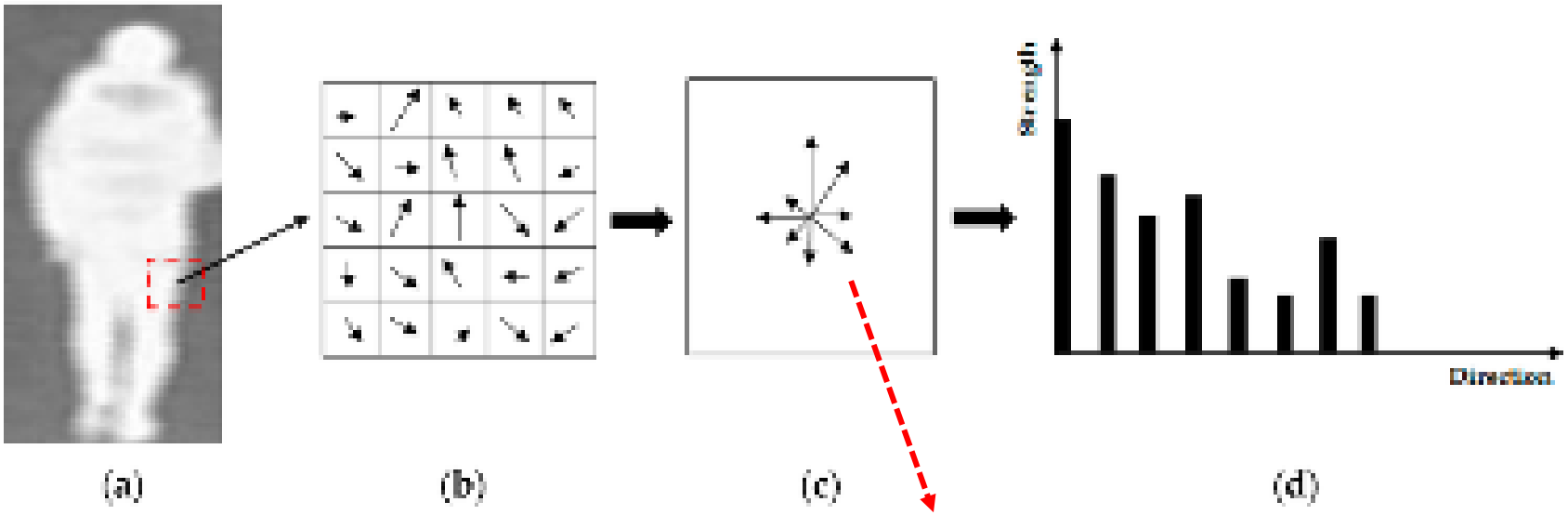
T. Ojala , M. Pietikainen , T. Maenpaa, " Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE T-PAMI, Aug. 2002

HOG: Histogram of Oriented Gradient

- 8-bin histogram
- Weighted by magnitude

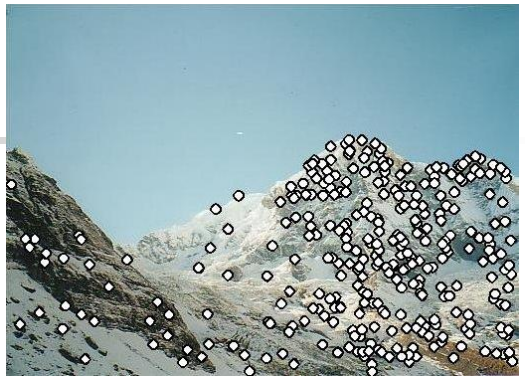
$$mag = \sqrt{\nabla_x^2 f(x,y) + \nabla_y^2 f(x,y)}$$

$$angle = \tan^{-1}\left[\frac{\nabla_y f(x,y)}{\nabla_x f(x,y)}\right]$$



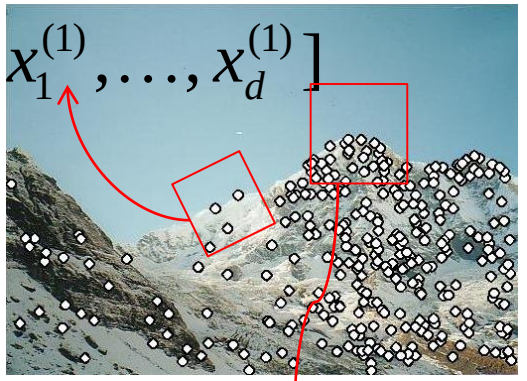
Do not count how many pixels has **direction k**, but instead sum the magnitudes of such pixels

Local Invariant Features: Scale Invariant Feature Transform (SIFT)



1) Detection: Identify the interest points (difference of Gaussian)

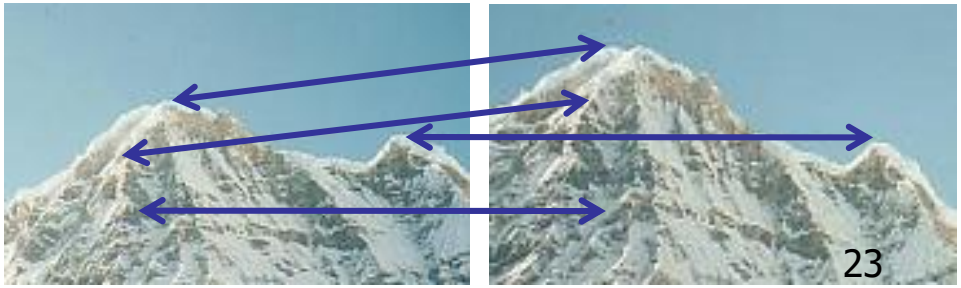
2) Description: Extract vector feature descriptor (HOG) surrounding each interest point.



$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views



Principal Component Analysis vs. Linear Discriminative Analysis

- Between-class (inter) scatter

$$S_B = \sum_{i=1}^{c} |\chi_i| (\mu_i - \mu)(\mu_i - \mu)^T$$

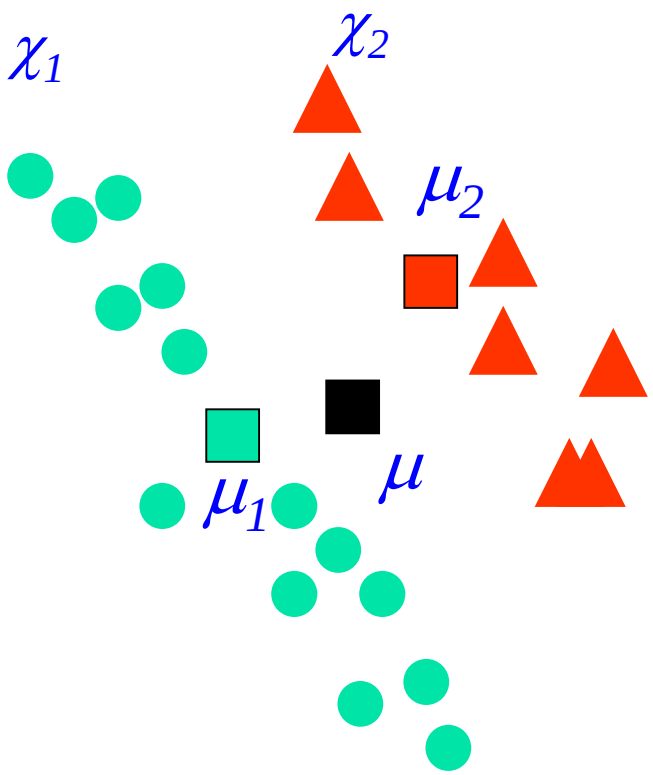
- Within-class (intra) scatter

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

- Total scatter (covariance matrix)

$$S_T = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu)(\mu_k - \mu)^T = S_B + S_W$$

  - c is the number of classes
  - μ_i is the mean of class χ_i
  - $|\chi_i|$ is number of samples of χ_i.



PCA vs. LDA

- PCA (orthonormal vector W)

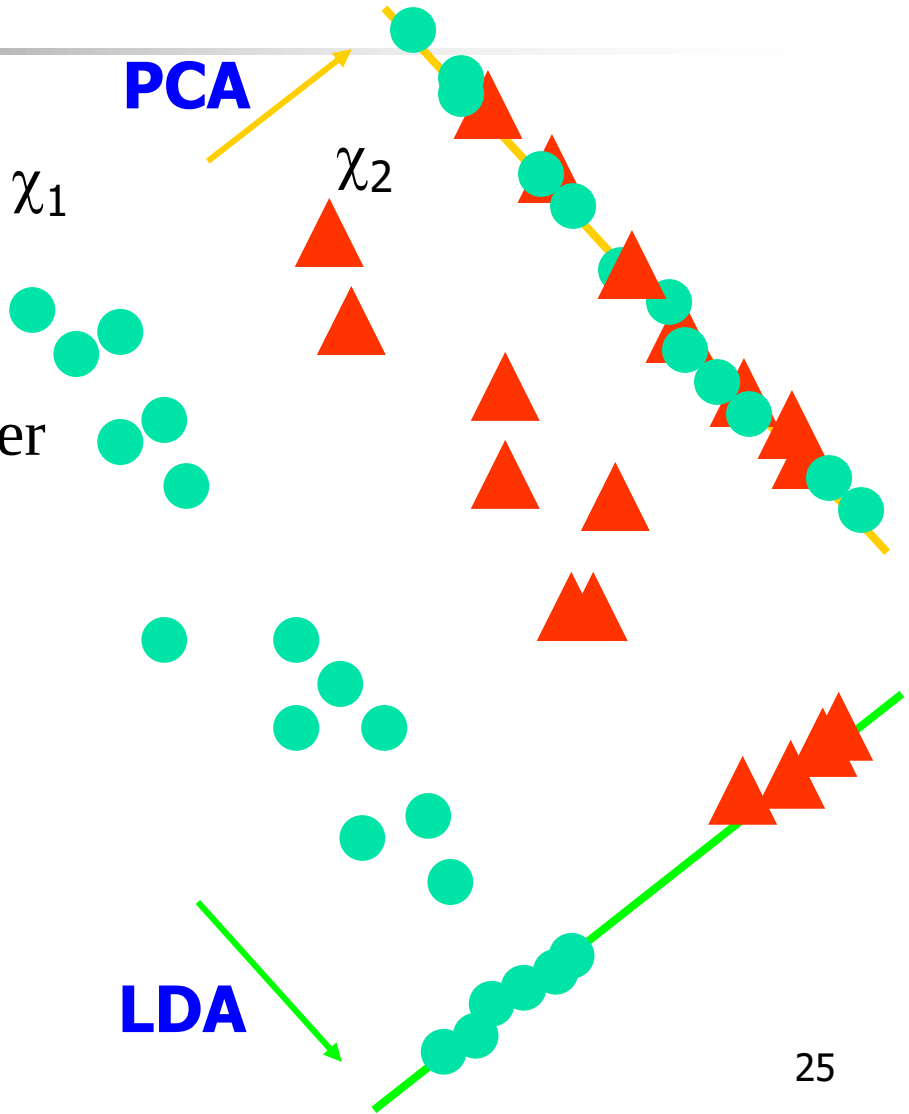
$$W_{PCA} = \arg\max_{W} \left| W^T S_T W \right|$$

Maximizes projected total scatter

- LDA

$$W_{LDA} = \arg\max_{W} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|}$$

Maximizes ratio of projected between-class to projected within-class scatter



Paradigms of Machine Learning

- Supervised Learning

Given $D = \{x_i, y_i\}$; Learn $y_i = f(x_i)$; s.t., $x_{new} \rightarrow y_{new}$

- Unsupervised Learning

Given $D = \{x_i\}$; Learn $y_i = f(x_i)$; s.t., $x_{new} \rightarrow y_{new}$

- Semi-Supervised (Inductive) Learning

Given $D = \{x_i, y_i\}$ and $\{x_j^u\}$; Learn $y_i = f(x_i)$; s.t., $x_{new} \rightarrow y_{new}$

- Reinforcement Learning (game, control, operation)

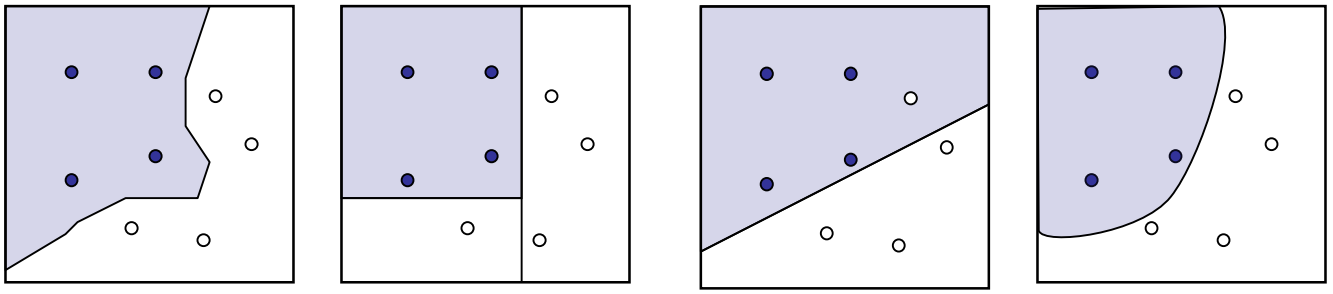
$\{s_t, a_t, s_{t+1}\} \rightarrow r_t$ (under a policy); choose $\{a_1, a_2, ..., a_T\}$; s.t., $\max \sum_{t=1}^{T} r_t$

- Query (Active) Learning

Given $D = \{x_i, y_i\}$ and $\{x_j^u\}$; Query $x_j^u \rightarrow y_j^*$; Learn $y = f(x)$; s.t., $x_{new} \rightarrow y_{new}$

Supervised Learning: Classification

- It can be arbitrary functions of high-dim $x \in R^m$



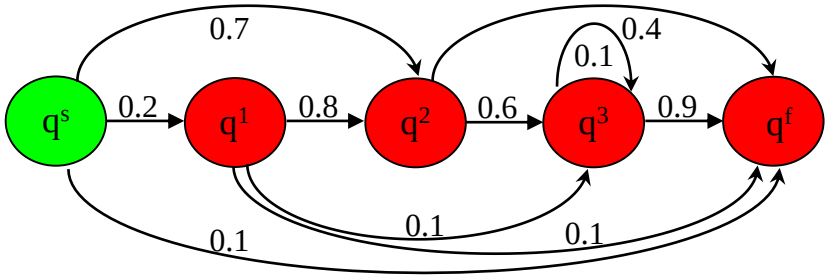
Nearest Neighbor (template matching)

Decision Tree (CART & RF)

Linear Functions (SVM & DML)

Nonlinear Functions (NN)

$\{x_1, x_2, x_3, ..., x_T\}$

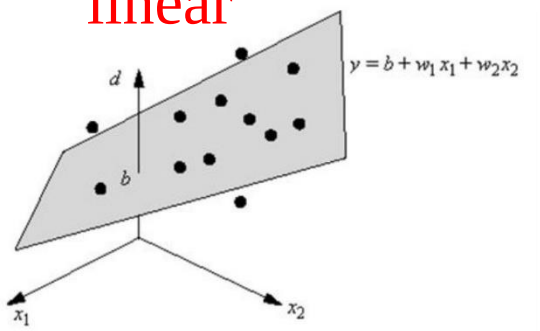


Hidden Markov Model (HMM)

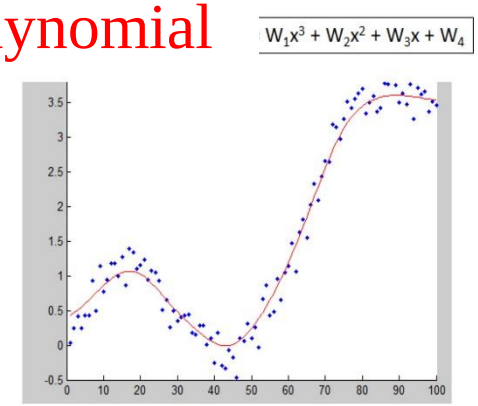
Supervised Learning: Regression

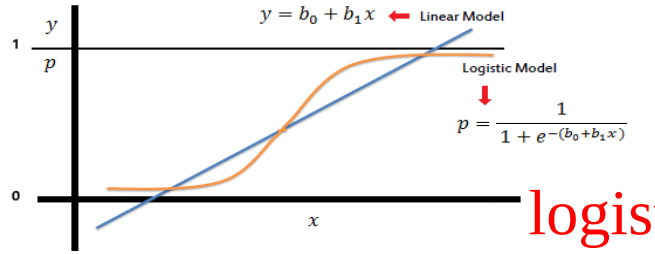
- A regressor models the relationship between a certain number of features and a continuous target variable.

linear



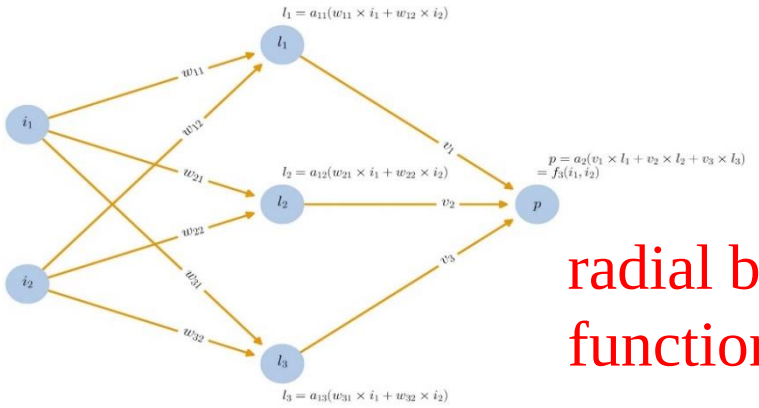
Polynomial



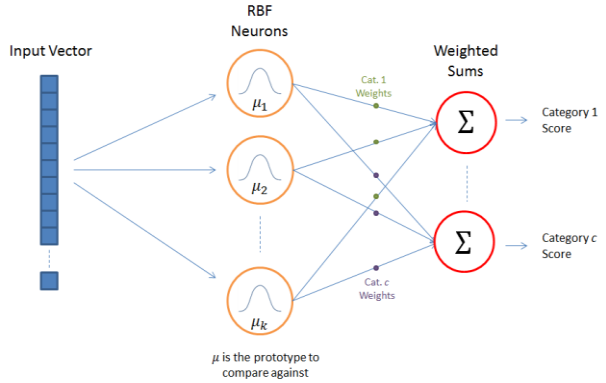


logistic

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots\beta_n X_n)}} = \frac{1}{1+e^{-(\beta_0+\sum \beta_i X_i)}}$$



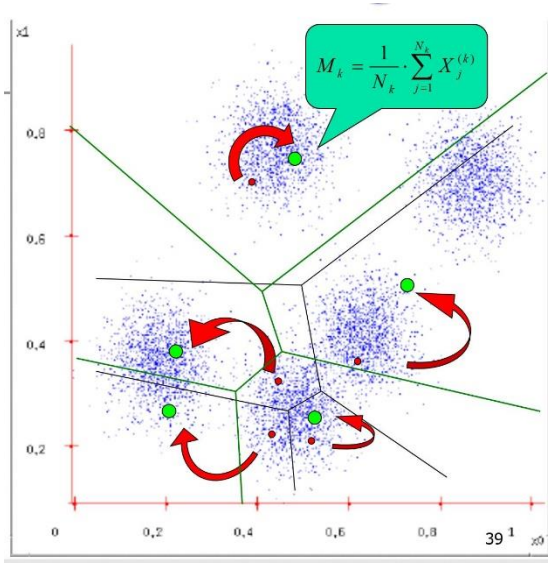
radial basis function

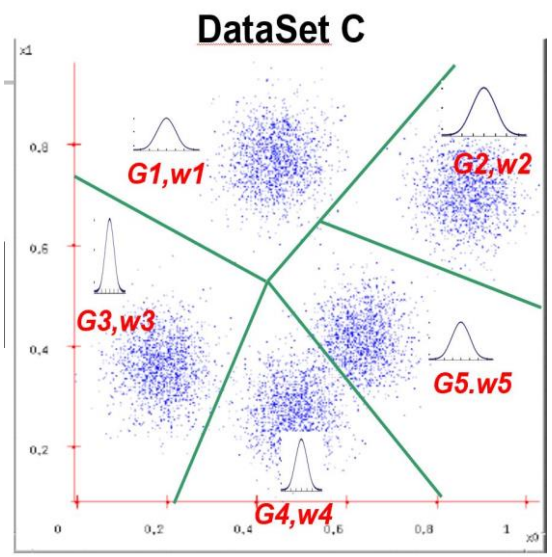


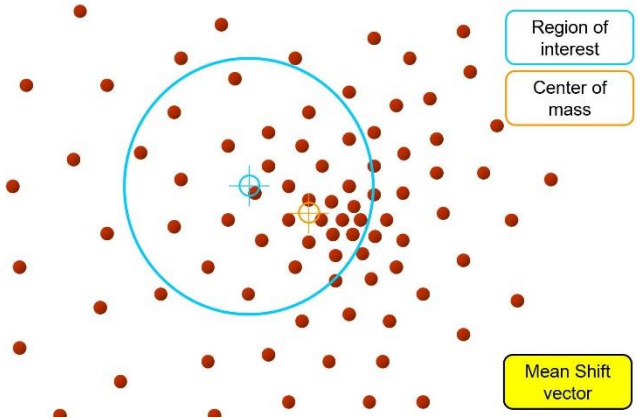
neural network

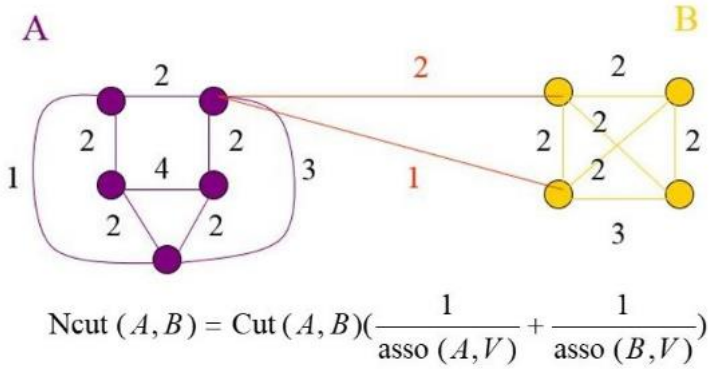
Unsupervised Learning: Clustering

- K-Mean
- Gaussian Mixture
- Mean Shift
- Normalized Cut



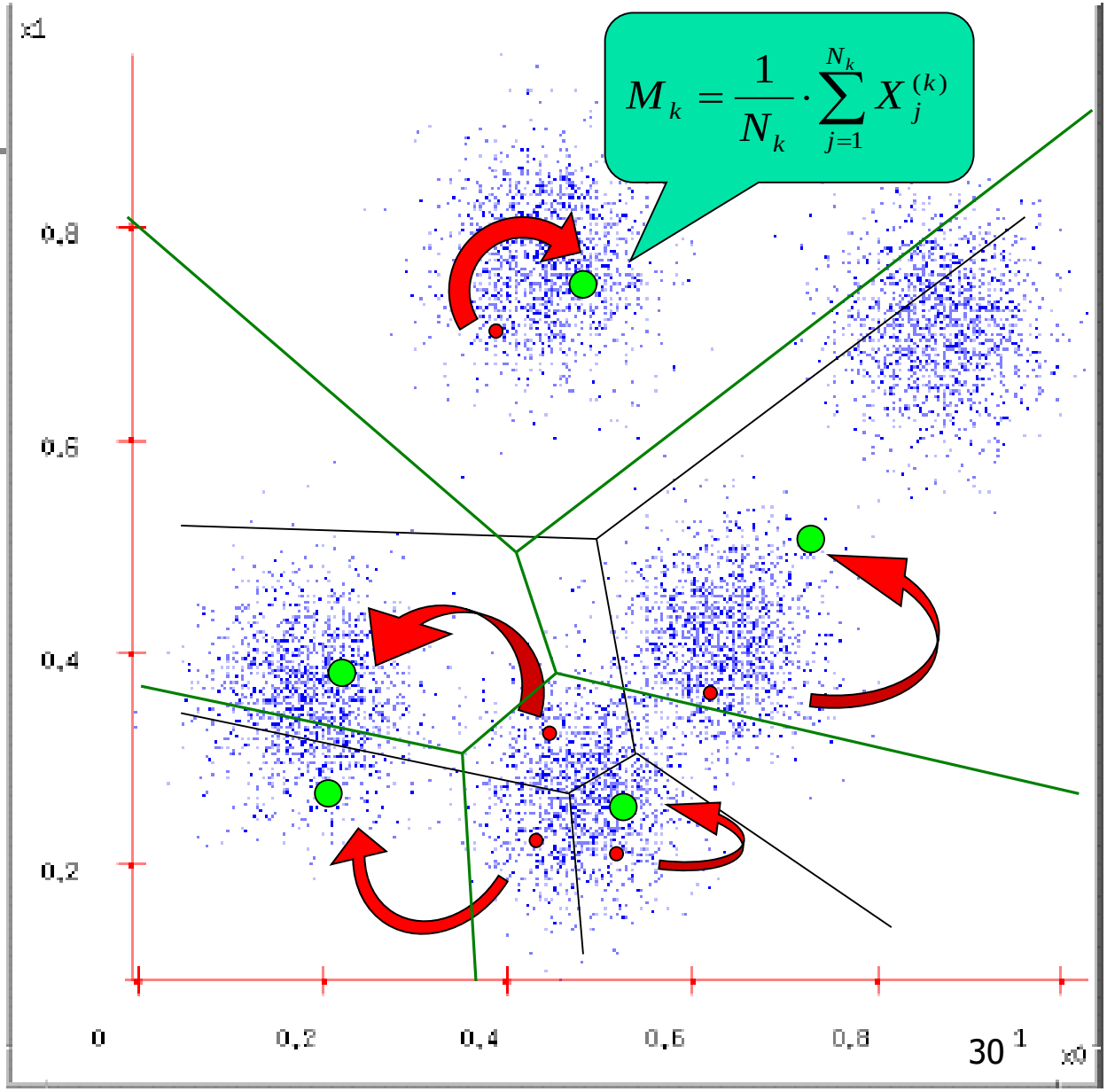






K-Means Clustering

1. How many clusters to be chosen.(e.g., $K=5$).
2. Randomly select *k* cluster centers locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)
4. Each cluster fine the centroid of the points it owns.
5. …and jumps to there
6. …repeat step 3 to 5 until terminated (little changes)



Splitting & Merging

Splitting

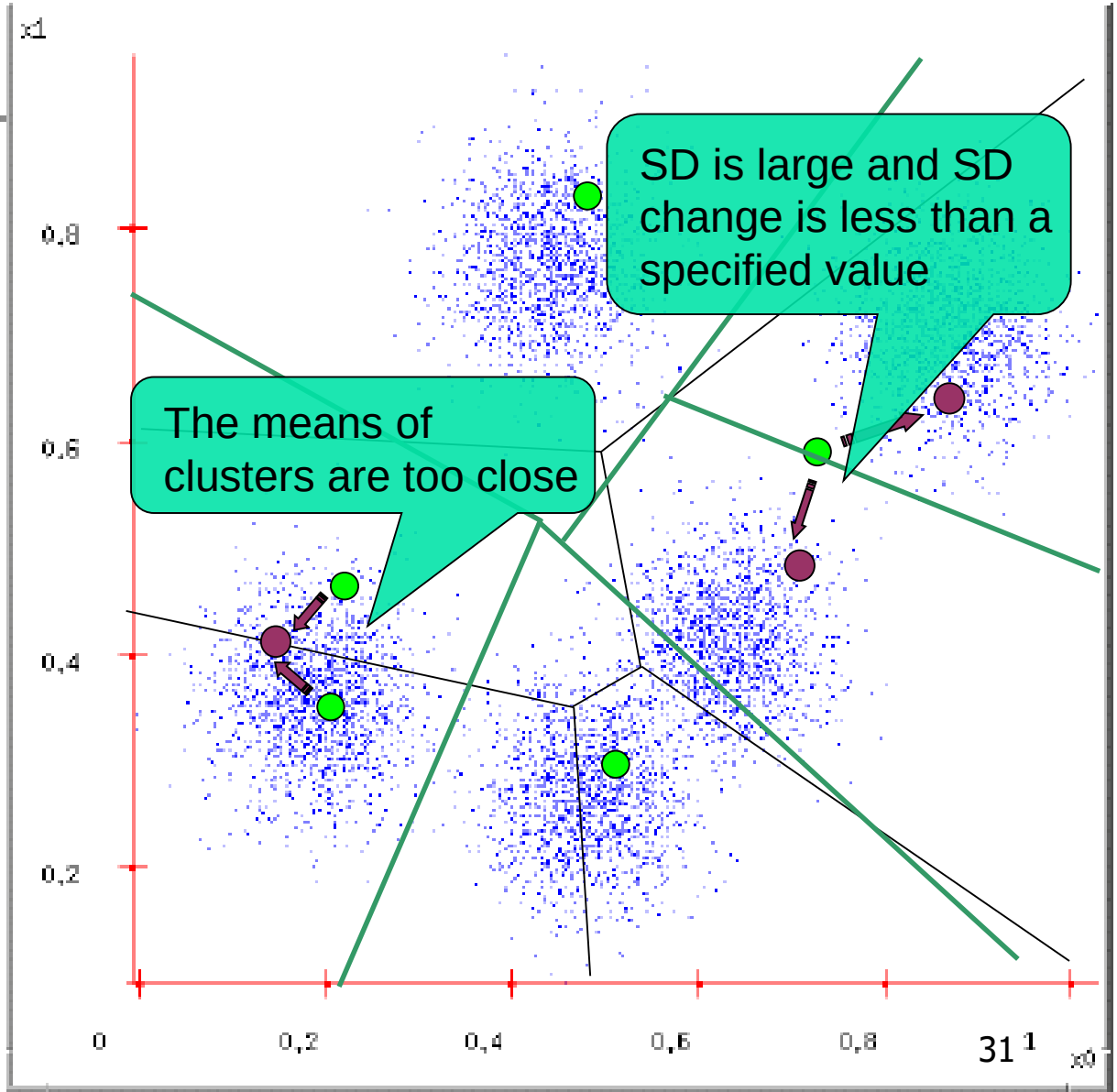
$$SD_k : \sigma_k = \sqrt{\sum_{j=1}^{N_k} \| X_j^{(k)} - M_k \|^2}$$

Merging

$$M_k = \frac{1}{N_k} \cdot \sum_{j=1}^{N_k} X_j^{(k)}$$

$$Dist(X, Y) = \sqrt{\sum_{i=1}^{m} (X_i - Y_i)^2}$$

Loop until the change of SD in all clusters are less than a specified value by user, or when a specified number of epochs have been reached.

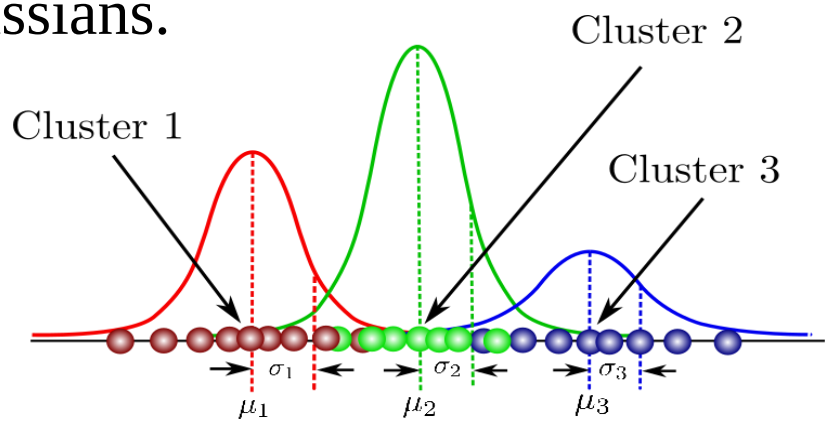


Gaussian Mixture Model (GMM)

- The Gaussian mixture algorithm estimates probability density functions (PDF) for a data set $\{X \in R^m\}$ based on a weighted average of multiple Gaussians.

$$P(X) = \sum_{k=1}^{K} w_k G_k(X)$$

Gaussian mixture as a universal functional approximation



where w_k is the weight of the k-th Gaussian G_k and the weights sum to one. One such PDF model is produced for each data class.

and $$G_k(X) = \frac{1}{(2\pi)^{m/2} |V_k|^{1/2}} \cdot e^{[-1/2(X-\mu_k)^T V_k^{-1}(X-\mu_k)]}$$

where μ_k is the mean and V_k is the covariance matrix of the Gaussian.

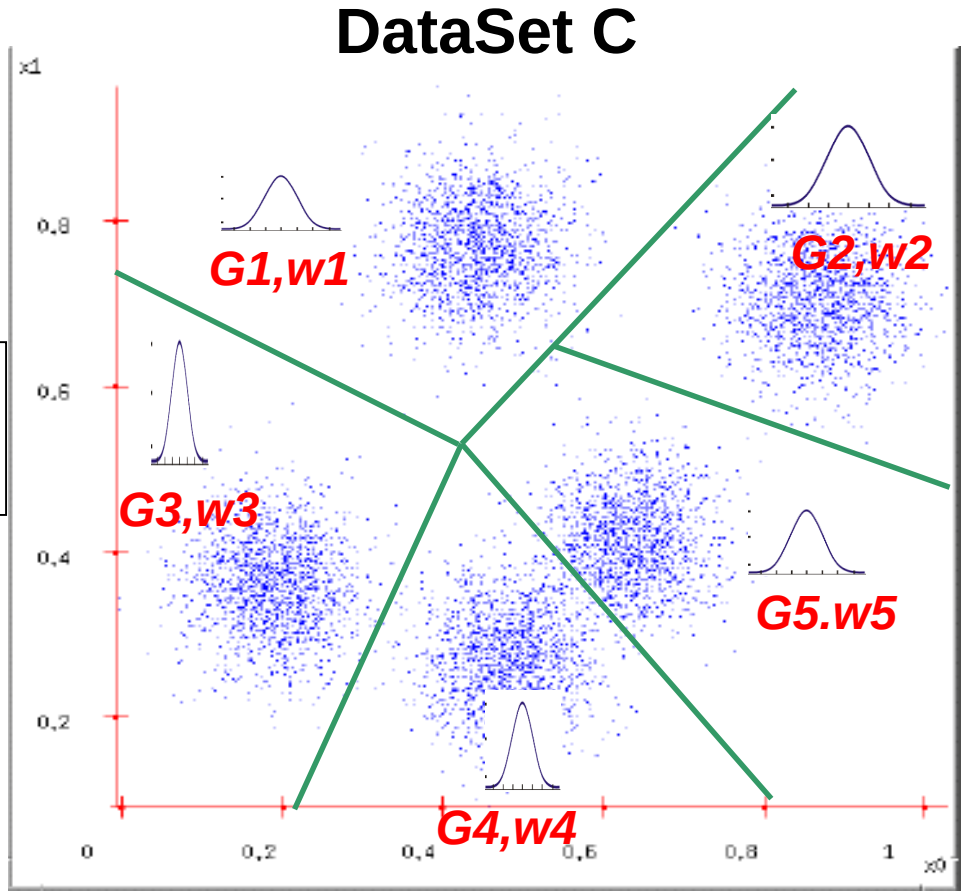
GMM Formulation for Unsupervised Data Clustering

$$P(X \mid C) = \sum_{k=1}^{K} w_k G_k$$

$$G_k \equiv p(X \mid G_k) = \frac{1}{(2\pi)^{m/2} \mid V_k \mid^{1/2}} \cdot e^{[-1/2(X-\mu_k)^T V_k^{-1}(X-\mu_k)]}$$

Variables: μ_k, V_k, w_k

EM (expectation-maximize) algorithm is used to approximate these variables.



GMM Training: EM Algorithm

- Initialize the initial **Gaussian means μ_k**, $k=1,\ldots K$ using the K-means clustering algorithm
- Initialize the **covariance matrices, V_k** , to the distance to the nearest cluster.
- Initialize the **weights $w_k = 1 / K$** so that all Gaussian are equally likely.

Present each pattern X of the data set C and model this data set a weighted sum of Gaussians:

$$p(X \mid C) = \sum_{k=1}^{K} w_k \, p(X \mid G_k)$$

where ***K*** is the number of Gaussians, the w_k's are the weights, and

$$p(X \mid G_k) = \frac{1}{(2\pi)^{m/2} \mid V_k \mid^{1/2}} \cdot e^{[-1/2(X-\mu_k)^T V_k^{-1}(X-\mu_k)]}$$

where V_k is the covariance matrix.

GMM Training

Expectation

Compute:

$$w_{kp} \equiv P(G_k \mid X_p) = \frac{w_k\, p(X_p \mid G_k)}{p(X)} = \frac{w_k\, p(X_p \mid G_k)}{\sum_{j=1}^{K} w_j\, p(X_p \mid G_j)}$$

Iteratively update the **weights**, **means** and **covariances**:

$$w_k(t+1) = \frac{1}{N}\sum_{p=1}^{N} w_{k\,p}(t)$$

Maximization

$$\mu_k(t+1) = \frac{1}{N\ w_k(t)}\sum_{p=1}^{N} w_{k\,p}(t) X_p$$

$$V_k(t+1) = \frac{1}{N\ w_k(t)}\sum_{p=1}^{N} w_{k\,p}(t)((X_p - \mu_k(t))(X_p - \mu_k(t))^T)$$

Supervised Learning

- Classification Trees and Random Forest
- Linear and Polynomial Regression
- Support Vector Machine for Classification

Decision Tree Classifer

- Decision trees are individual attribute learners that are combined. They are one of the most popular learning methods commonly used for data exploration.
- One type of decision tree is called CART -- classification and regression tree -- *L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, 1984, CRC press*
- CART … greedy, top-down binary, recursive partitioning, that divides feature space into sets of disjoint rectangular regions.

Recursive Partitioning

- Pick one of the *m*-dim predictor variables, $x_i, i = 1,2,...,m$
- Pick a value of this *dimen*, say s_i, that divides the training data into two (not necessarily equal) portions
- Measure how "pure" or "homogeneous" each of the resulting portions are

  "Pure" = containing records of mostly one class
- Try different values of $\{x_i, s_i\}$ to maximize purity in initial split (or minimize impurity)
- After you get a "maximum purity" split, repeat the process for a second split, and so on (assume *K* classes)

ImPurity: Gini Index or Entropy

$$I(A) = 1 - \sum_{k=1}^{K} p_k^2$$

$$entropy(A) = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

- Gini Index for set *A* containing *m* records
- *p* = proportion of cases in set *A* that belong to class *k*
  - *I(A) = 0* when all cases belong to same class
  - Max value when all classes are equally represented (= 0.50 in binary case)
- *p* = proportion of cases (out of *m*) in set *A* that belong to class *k*
- Entropy ranges between 0 (most pure) and $\log_2(K)$ (equal representation of classes)

Example: Riding Mowers

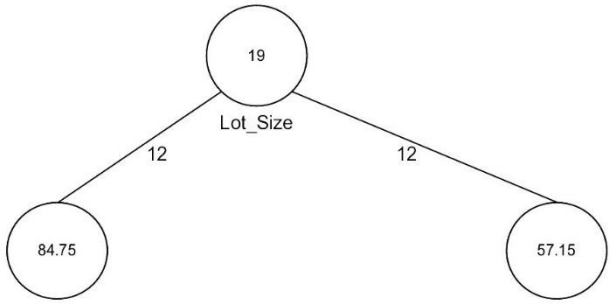
Goal: Classify 24 households as owning or not owning riding mowers

Predictors = Income, Lot Size

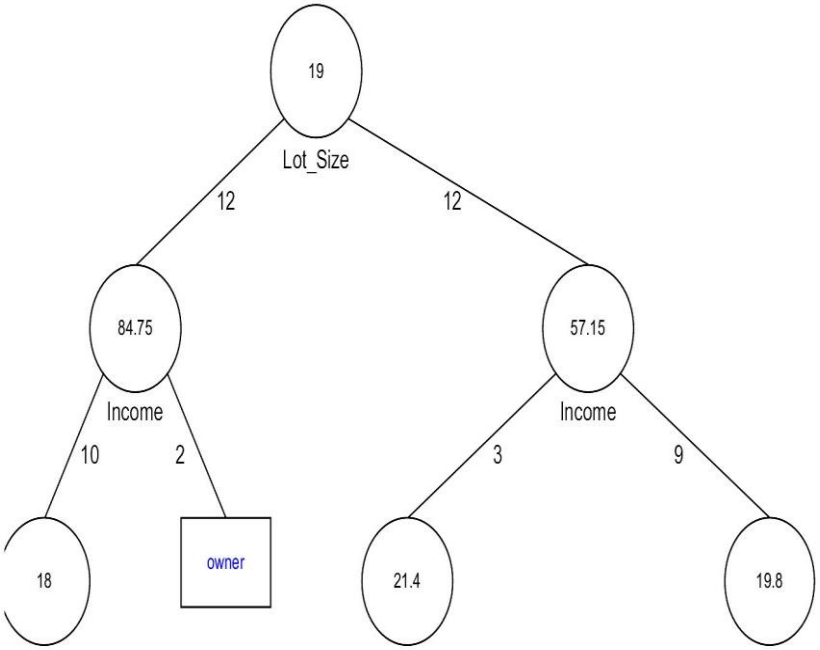
Income	Lot_Size	Ownership
60.0	18.4	owner
85.5	16.8	owner
64.8	21.6	owner
61.5	20.8	owner
87.0	23.6	owner
110.1	19.2	owner
108.0	17.6	owner
82.8	22.4	owner
69.0	20.0	owner
93.0	20.8	owner
51.0	22.0	owner
81.0	20.0	owner
75.0	19.6	non-owner
52.8	20.8	non-owner
64.8	17.2	non-owner
43.2	20.4	non-owner
84.0	17.6	non-owner
49.2	17.6	non-owner
59.4	16.0	non-owner
66.0	18.4	non-owner
47.4	16.4	non-owner
33.0	18.8	non-owner
51.0	14.0	non-owner
63.0	14.8	non-owner

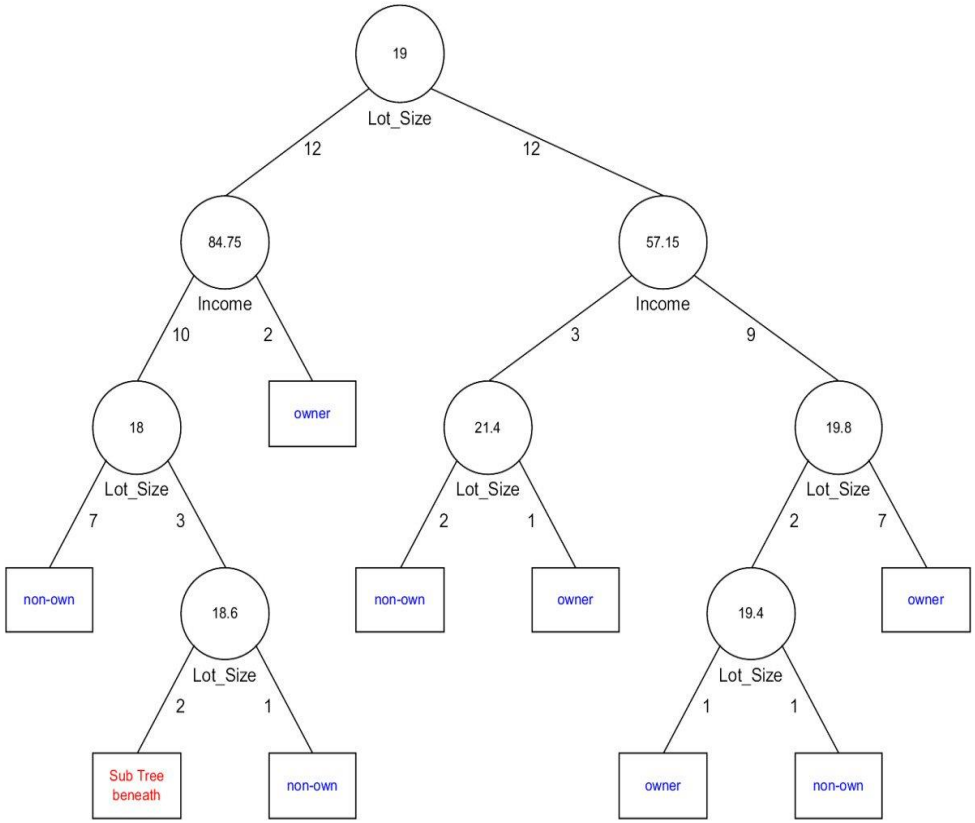
Impurity and Recursive Partitioning

- Obtain overall impurity measure (weighted avg. of individual sets)
- At each successive stage, compare this measure across all possible splits (depends on the steps) in all variables
- Choose the split that reduces impurity the most
- Chosen split points become nodes on the tree

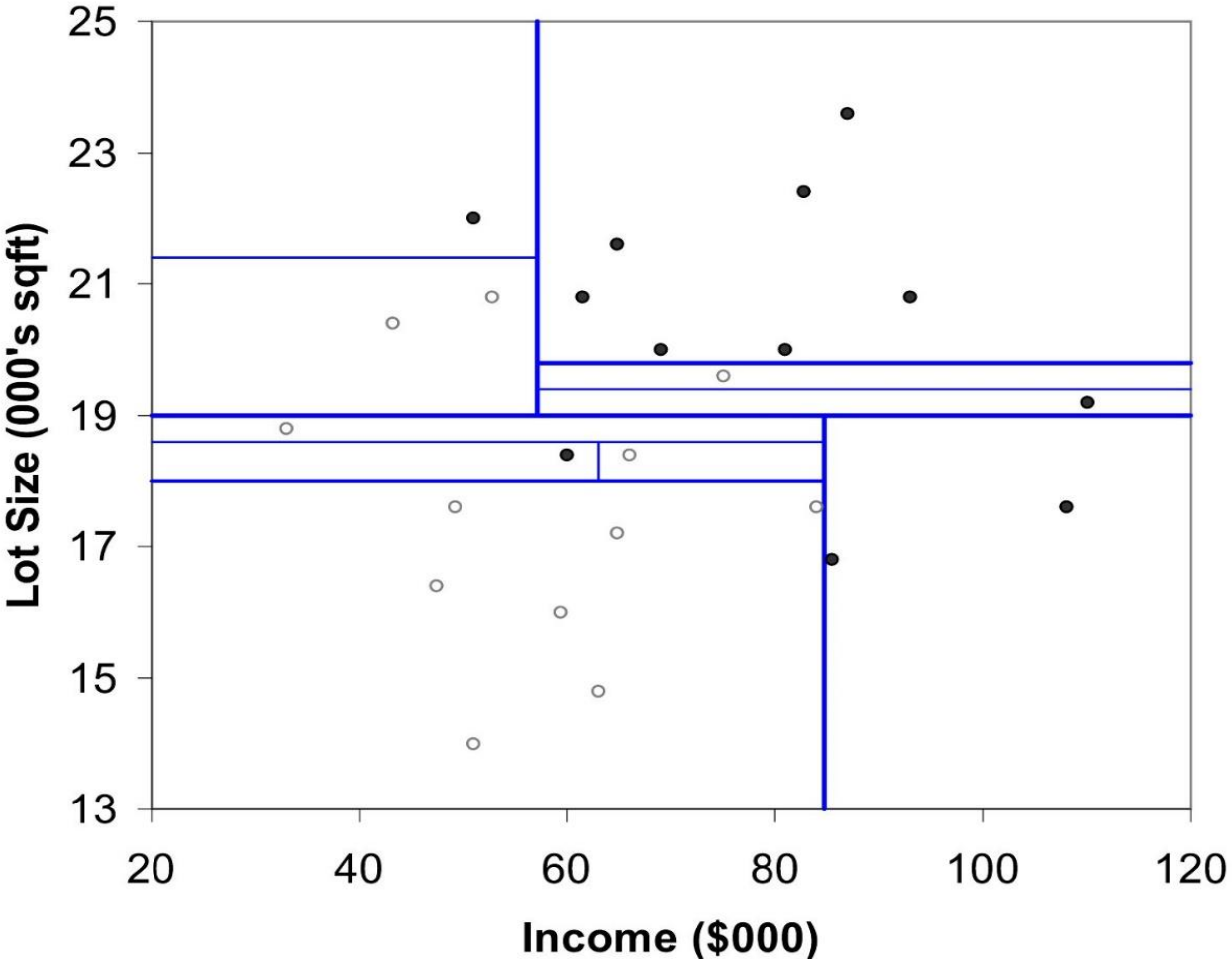


Intermediate Tree and Final Tree

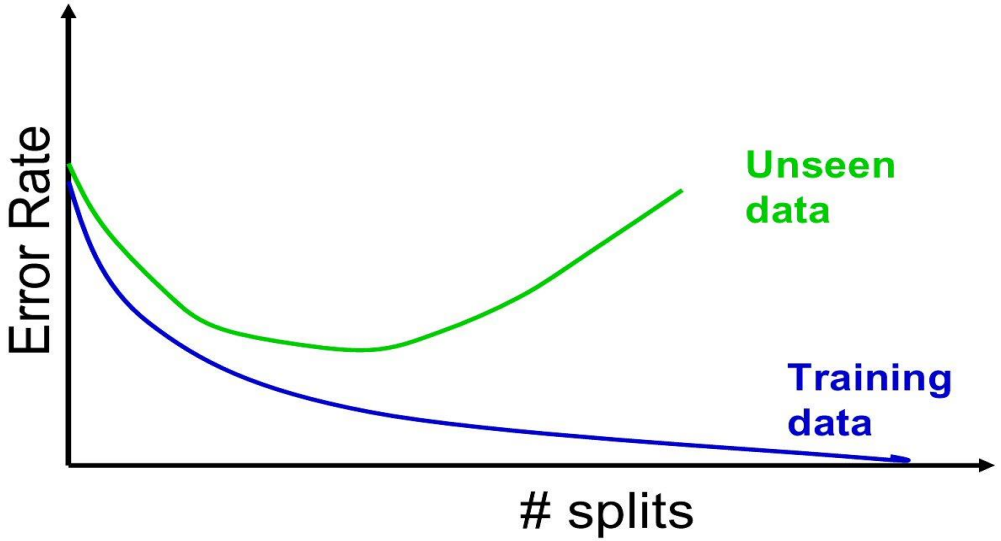




After All Splits



Cross Validation and Tree Pruning



- Divide the data into two sets: Training and Unseen
- CART lets tree grow to full extent, then prunes back
- Idea is to find that point at which the validation error begins to rise

Random Forest (RF)

- Suppose our training data set is represented by T and suppose data set has m features (or attributes or variables).

$T = \{(X_1,y_1), (X_2,y_2), \ldots (X_N, y_N)\}$ and $X_i \in R^m$

X_i is input vector $\{x_{i1}, x_{i2}, \ldots x_{im}\}$, y_i is the class label (or output or class).

- Suppose we decide to have s number of trees in our forest then we first create R datasets of "same size as original" created from "random resampling" of data in T with-replacement (N times for each dataset). This will result in $\{T_1, T_2, \ldots T_R\}$ datasets.
- Each of these is called a bootstrap dataset. Due to "with-replacement" every dataset T_r can have duplicate data records and T_r can be missing several data records from original datasets. This is called Bootstrapping.

Random Forest (RF)

- Bagging is the process of taking bootstraps & then aggregating the models learned on each bootstrap.
- Now, RF creates R trees and uses p (=sqrt(m) or =floor(ln(m)+1)) random subfeatures out of m possible features to create any tree. This is called random subspace method.
- So for each T_r bootstrap dataset you create a tree G_r. If you want to classify a new input data $D \in R^m$, you let it pass through each tree and produce R outputs (one for each tree) which can be denoted by $Y = \{y_1, y_2, ..., y_R\}$.
- Final prediction is a majority vote on this set.
- [Andy Liaw and Matthew Wiener, R-News, December 2002]

Summaries of Random Forest

Leo Breiman, "Random Forests," Machine Learning, 45(1), 2001

- Fast fast fast!
  - RF is fast to build. Even faster to predict! Resistance to over training
  - Practically speaking, not requiring cross-validation for model selection significantly speeds training by *10x-100x* or more. Can be fully parallelizable -> even faster!
- Ability to handle data without preprocessing
  - data does not need to be rescaled, transformed, or modified (also true for CART)
  - resistant to outliers & automatic handling of missing values

Linear and Polynomial Regression

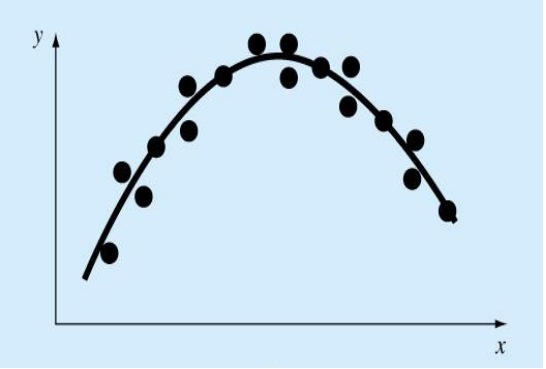
Polynomial Regression

- The linear regression model $y = \boldsymbol{a}^T\boldsymbol{x}+e$ is a general model for fitting any relationship that is linear with the unknown parameter $\boldsymbol{a}$.
- Polynomial regression model (curve fitting):
  - 2nd order polynomial of one variable

$$y = a_o + a_1 x + a_2 x^2 + e$$

  - 2nd order polynomial of two variables

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_{11} x_1^2 + a_{22} x_2^2 + a_{12} x_1 x_2 + e$$



Polynomial Regression Fitting (Learning)

- **A 2nd order polynomial (quadratic)** is defined by:

$$y = a_o + a_1 x + a_2 x^2 + e$$

- The residuals between the model and the data:

$$e_i = y_i - a_o - a_1 x_i - a_2 x_i^2$$

- Minimize the sum of squares of the residual (given N data pairs $\{(x_i, y_i)\}$) – least squares solution

$$S_r = \sum e_i^2 = \sum \left(y_i - a_o - a_1 x_i - a_2 x_i^2 \right)^2$$

Polynomial Regression Fitting (Learning)

$$\frac{\partial S_r}{\partial a_o} = -2\sum (y_i - a_o - a_1 x_i - a_2 x_i^2) = 0$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum (y_i - a_o - a_1 x_i - a_2 x_i^2) x_i = 0$$

$$\frac{\partial S_r}{\partial a_2} = -2\sum (y_i - a_o - a_1 x_i - a_2 x_i^2) x_i^2 = 0$$

$$\sum y_i = n \cdot a_o + a_1 \sum x_i + a_2 \sum x_i^2$$

$$\sum x_i y_i = a_o \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3$$

$$\sum x_i^2 y_i = a_o \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4$$

3 linear equations with 3 unknowns (a_o,a_1,a_2), can be solved

Polynomial Regression Fitting (Learning)

- A system of *3x3* equations needs to be solved to determine the coefficients of the polynomial.
- Similarly can be extended to solving a *kxk* equation

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{Bmatrix}$$

$$\boldsymbol{Ba=c \quad \rightarrow \quad a^*=B^+c=(B^TB)^{-1}B^Tc}$$

A 2nd Order Regression Example

x_i	y_i	x_i^2	x_i^3	x_i^4	$x_i y_i$	$x_i^2 y_i$
0	2.1	0	0	0	0	0
1	7.7	1	1	1	7.7	7.7
2	13.6	4	8	16	27.2	54.4
3	27.2	9	27	81	81.6	244.8
4	40.9	16	64	256	163.6	654.4
5	61.1	25	125	625	305.5	1527.5
15	152.6	55	225	979	585.6	2489

$$\sum x_i = 15$$

$$\sum y_i = 152.6$$

$$\sum x_i^2 = 55$$

$$\sum x_i^3 = 225$$

$$\sum x_i^4 = 979$$

$$\sum x_i y_i = 585.6$$

$$\sum x_i^2 y_i = 2488.8$$

Similarily for $y = a_0 + a_1 x_1 + a_2 x_2 + a_{11} x_1^2 + a_{22} x_2^2 + a_{12} x_1 x_2 + e$

A 2nd Order Regression Example

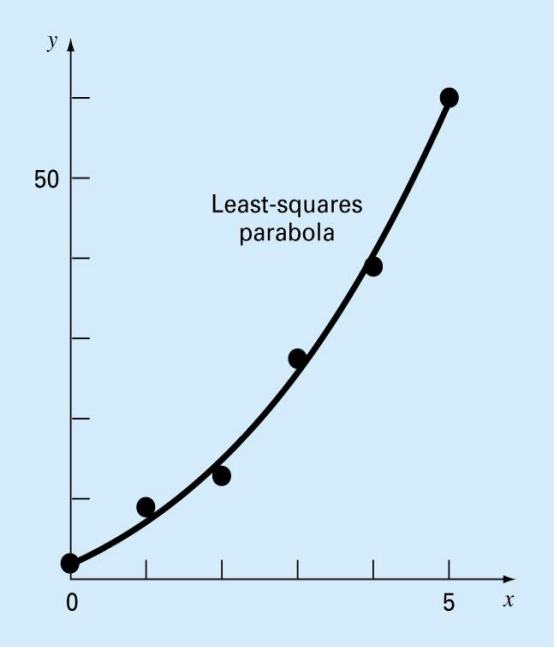
- The system of simultaneous linear equations:

$$\begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 152.6 \\ 585.6 \\ 2488.8 \end{Bmatrix}$$

$$a_0 = 2.47857,\ a_1 = 2.35929,\ a_2 = 1.86071$$

$$y = 2.47857 + 2.35929\,x + 1.86071\,x^2$$

$$S_r = \sum e_i^{\,2} = 3.74657$$



Support Vector Machine (SVM) Classifiers

Support Vector Machine (SVM)

- A classifier derived from statistical learning theory by Boser, Guyon, Vapnik, et al. in 1992
- SVM became famous when, using images as input, giving accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

B.E. Boser, I.M. Guyon, V.N. Vapnik, "A training algorithm for optimal margin classifiers," 5th annual workshop on Computational learning theory (COLT), 1992.

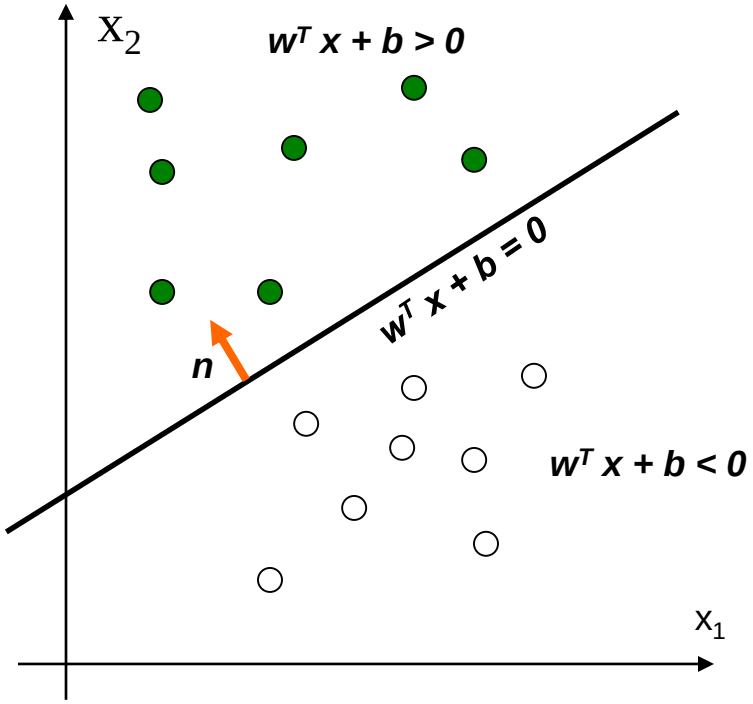
Linear Discriminant Function (Binary Classifer)

- $g(x)$ is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

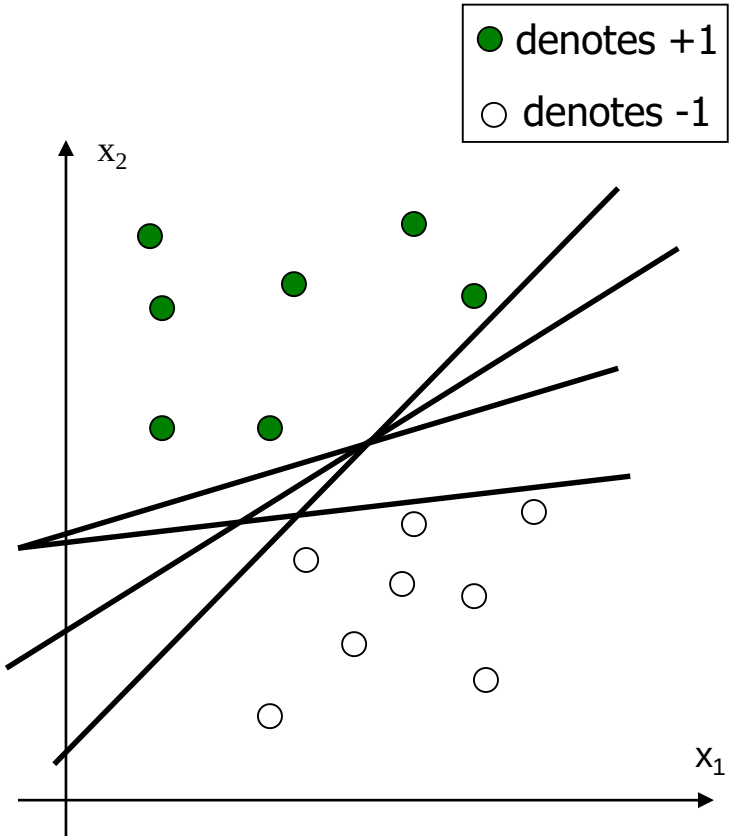
  - A hyper-plane in the feature space
  - (unit-length) normal vector of the hyper-plane:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



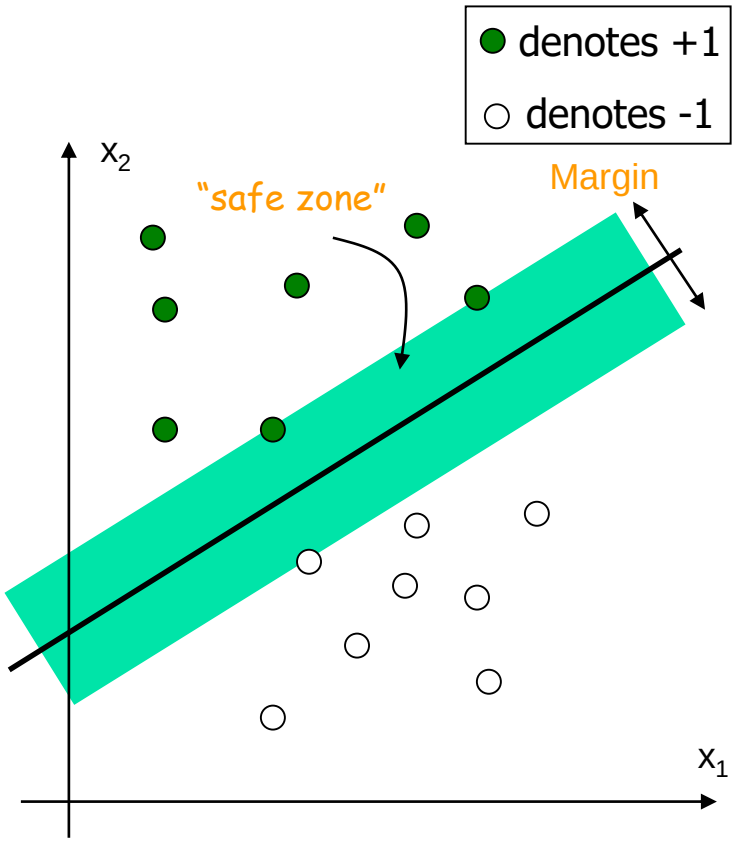
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
  - Infinite number of answers!
  - Which one is the best?



Large Margin Linear Classifier

- The linear discriminant function (classifier) with the maximum margin is the best
  - Margin is defined as the width that the boundary could be increased by before hitting a data point
  - Why it is the best?
    - Robust to outliners and thus strong generalization ability



Large Margin Linear Classifier

- Given a set of data points:

$\{(\mathbf{x}_i, y_i)\}, i = 1,2,...,n$, where

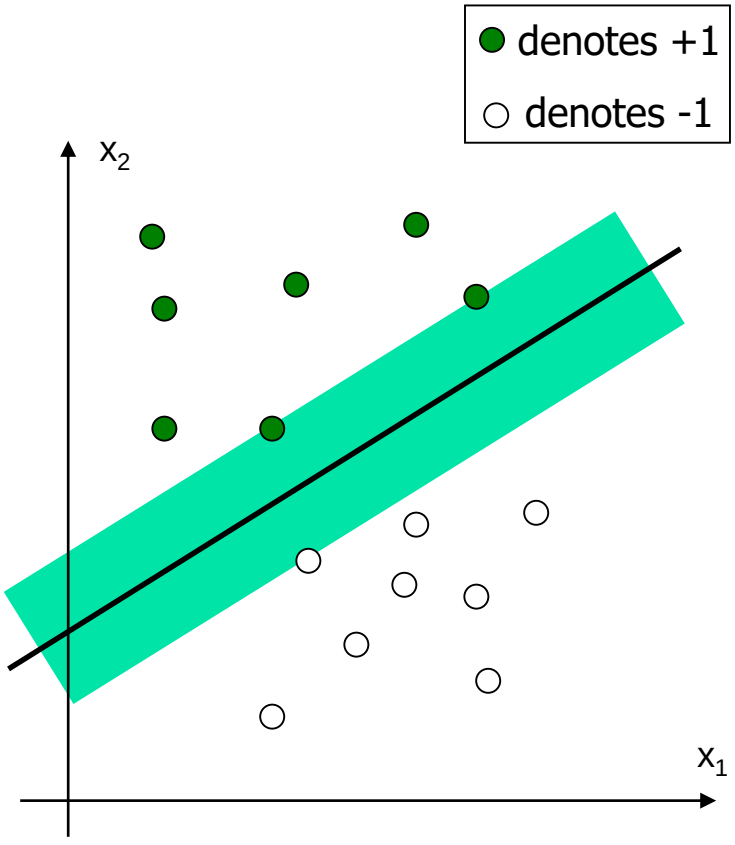
For $y_i = +1$, $\mathbf{w}^T\mathbf{x}_i + b > 0$

For $y_i = -1$, $\mathbf{w}^T\mathbf{x}_i + b < 0$

- With a scale transformation on both *w* and *b*, the above is equivalent to

For $y_i = +1$, $\mathbf{w}^T\mathbf{x}_i + b \geq 1$

For $y_i = -1$, $\mathbf{w}^T\mathbf{x}_i + b \leq -1$



Large Margin Linear Classifier

- Define support vectors $\{\mathbf{x}^+\}$ and $\{\mathbf{x}^-\}$

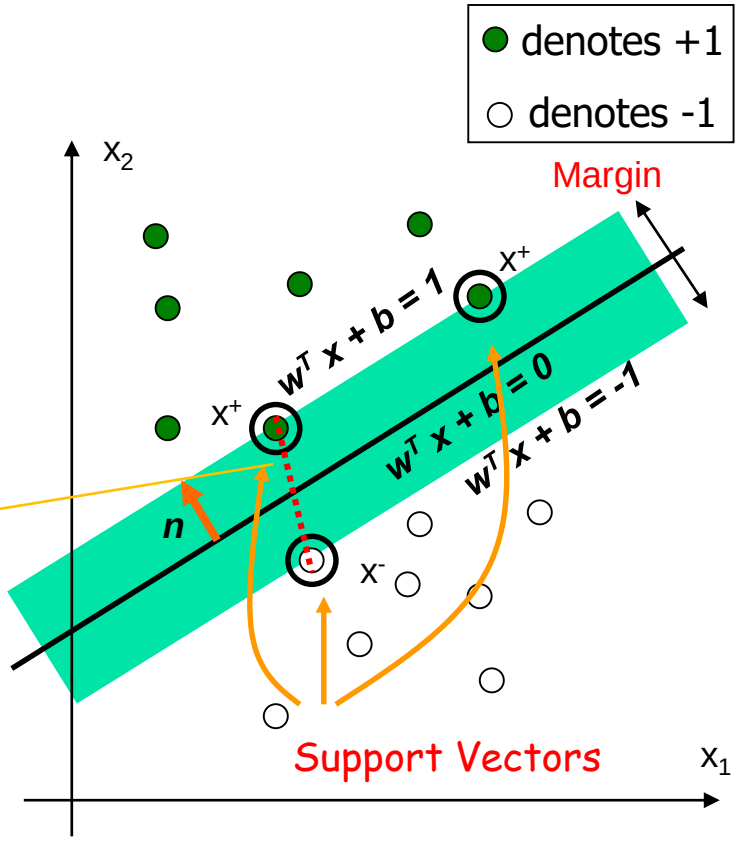
$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T\mathbf{x}^- + b = -1$$

  - The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-)\cdot\mathbf{n}$$

$$= (\mathbf{x}^+ - \mathbf{x}^-)\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$



Large Margin Linear Classifier

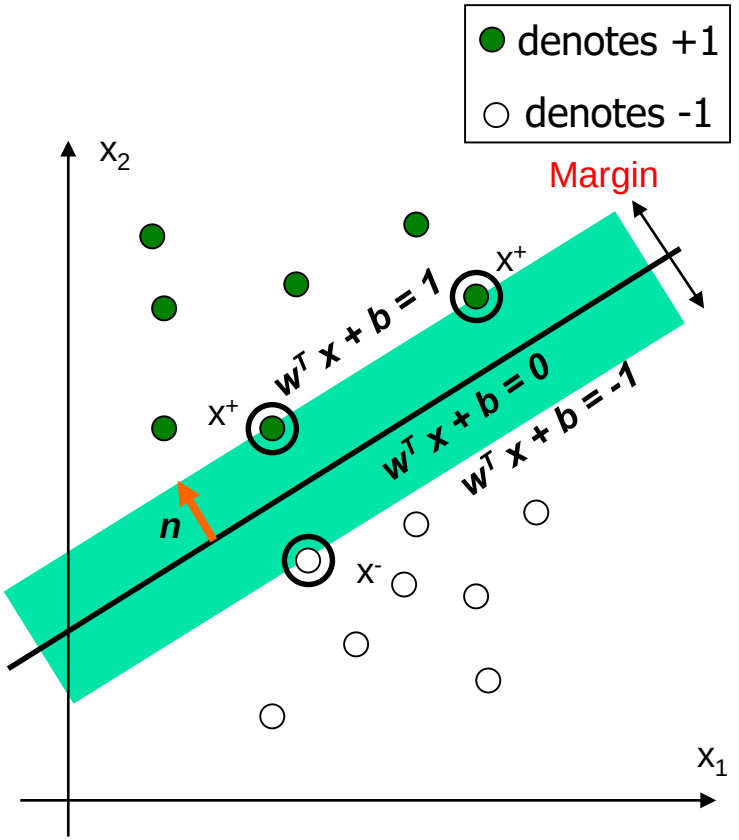
- Formulation: finding **w**, b

$$\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

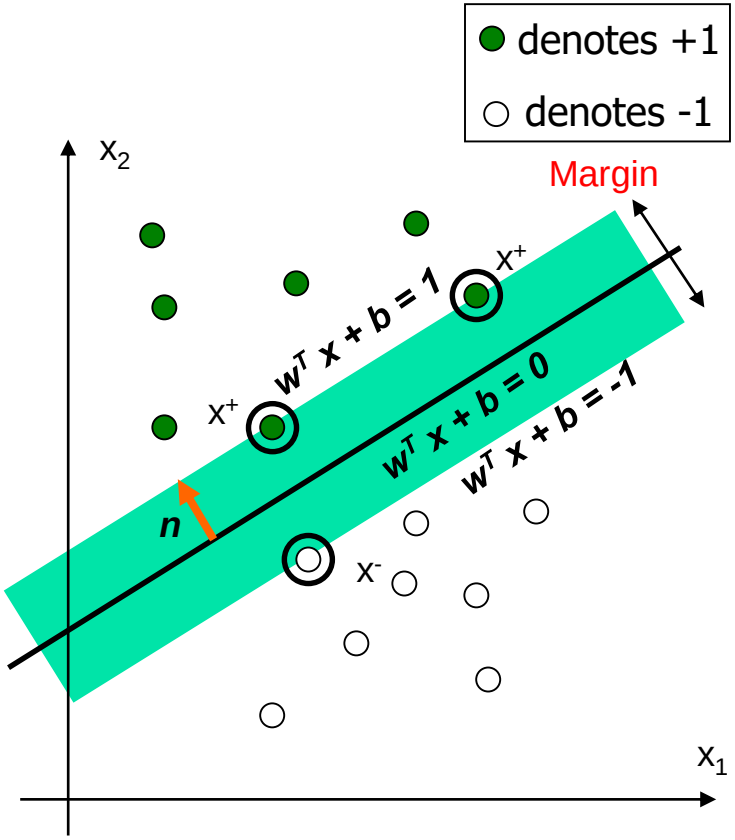
- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



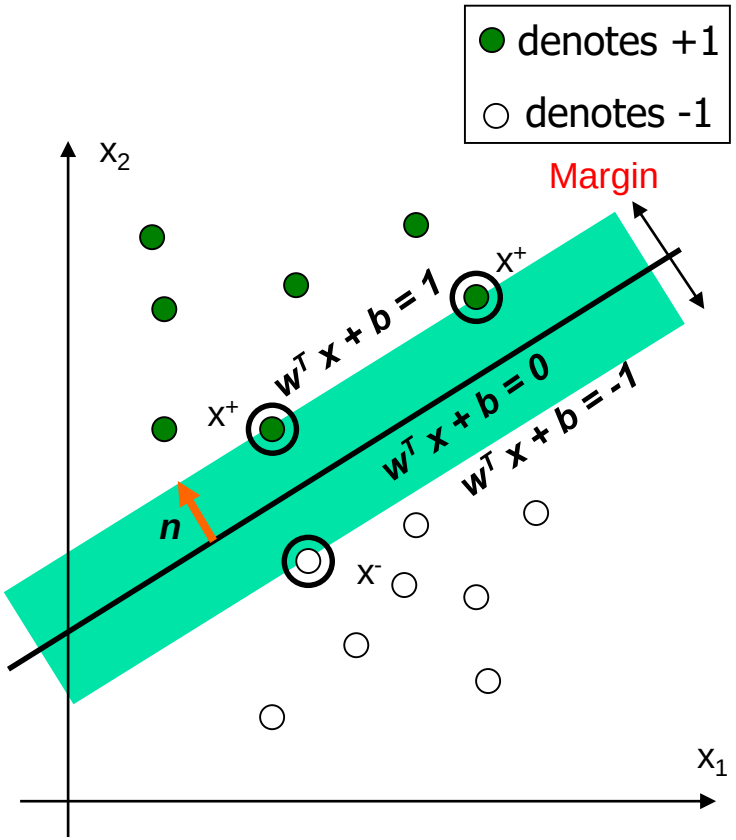
Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



Solving the Optimization Problem

Primal Problem

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w}$$

$$\text{s.t.,} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

Lagrangian Function

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.,} \quad \alpha_i \geq 0 \quad \forall i$$

Solving the Optimization Problem

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t., } \alpha_i \geq 0 \quad \forall i$$

$$\frac{\partial L_p}{\partial \mathrm{w}} = 0 \Rightarrow \mathrm{w} + \sum_{i=1}^{n} \alpha_i (-y_i) \mathrm{x}_i = 0 \Rightarrow \mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i \mathrm{x}_i$$

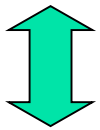
$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the Quadratic Programming Problem

$$L_p = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i y_i \mathrm{w}^{\mathrm{T}} \mathrm{x}_i - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i$$

$$\mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i \mathrm{x}_i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Lagrangian Dual Problem (solving α_i only)



Can be solved by a convex optimization

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

A quadratic function of α_i

$$\text{s.t.,} \quad \alpha_i \geq 0 \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

The Optimization Solution

- Obtaining a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, the solution to the primal is:

$$\mathrm{w} = \sum_{\mathrm{i} \in \mathrm{SV}} \alpha_i y_i x_i, \quad \text{and} \quad \mathrm{b} = \mathrm{y_k} - \sum_{\mathrm{i} \in \mathrm{SV}} \alpha_i y_i x_i^T x_k, \quad \text{for any } \alpha_\mathrm{k} > 0$$

where each non-zero α_i → $\mathbf{x}_i$ is a support vector.

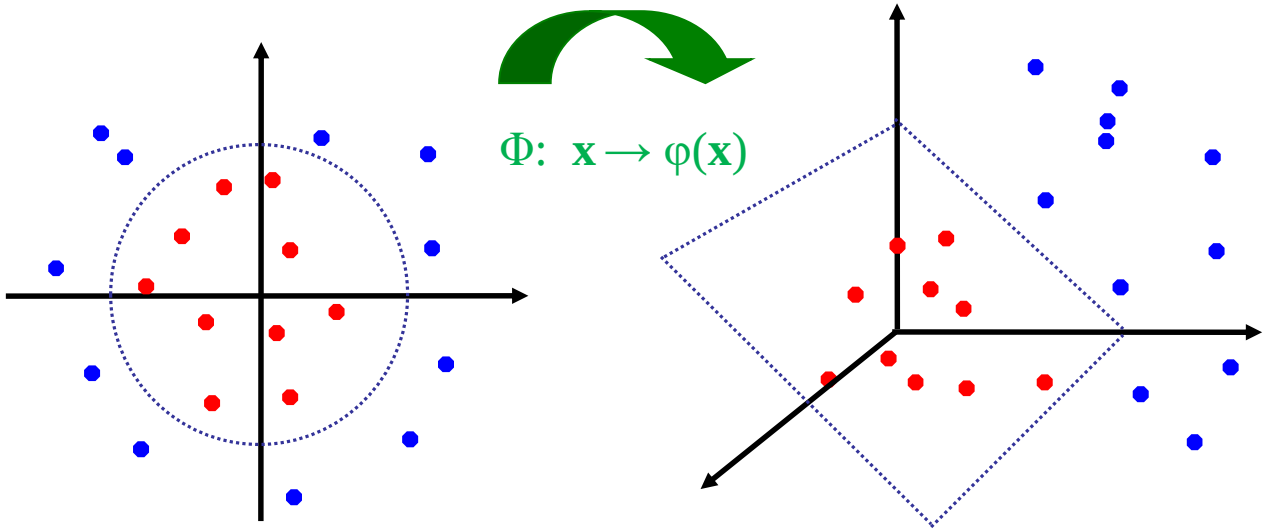
- The linear discriminant function (the SVM classification function) can thus be:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i \in \mathrm{SV}} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice it relies on a *dot product* between the test point **x** and the support vectors $\mathbf{x}_i$
- Also, solving the optimization problem involved computing the dot products $\mathbf{x}_i^T \mathbf{x}_j$ between all pairs of training points

Non-linear SVMs: Feature Space

- General idea: the original input space can be mapped to some "implicitly" higher-dimensional feature space (via nonlinear mapping) where the training set is more separable:



Nonlinear SVMs: The Kernel Trick

- With this mapping, the discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \text{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b$$

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions (positive definite):
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$
  - Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

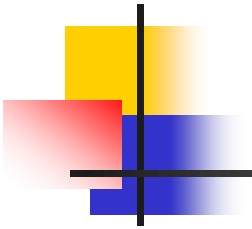
$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{such that} \quad 0 \le \alpha_i \le C \qquad \sum_{i \in SV} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \mathrm{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

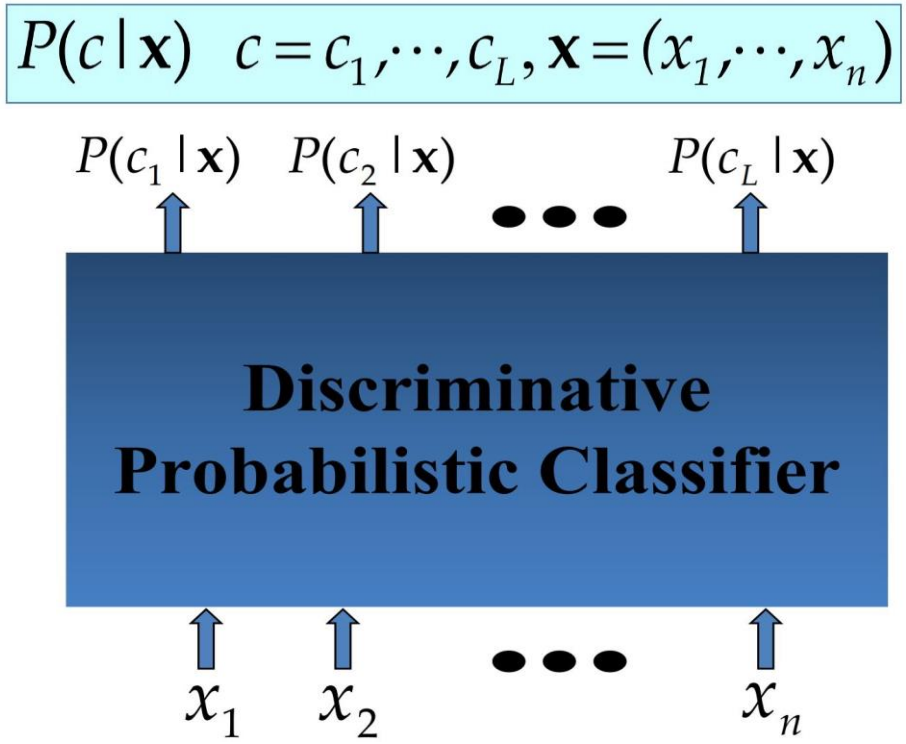
- The optimization technique is the same.



Modeling the Probabilities of Classification

Discriminative Classifiers

- Discriminative Classifiers: all training examples of different classes must be jointly used to build up a single discriminative classifier, e.g., CART, SVM, MLP, CNN, etc



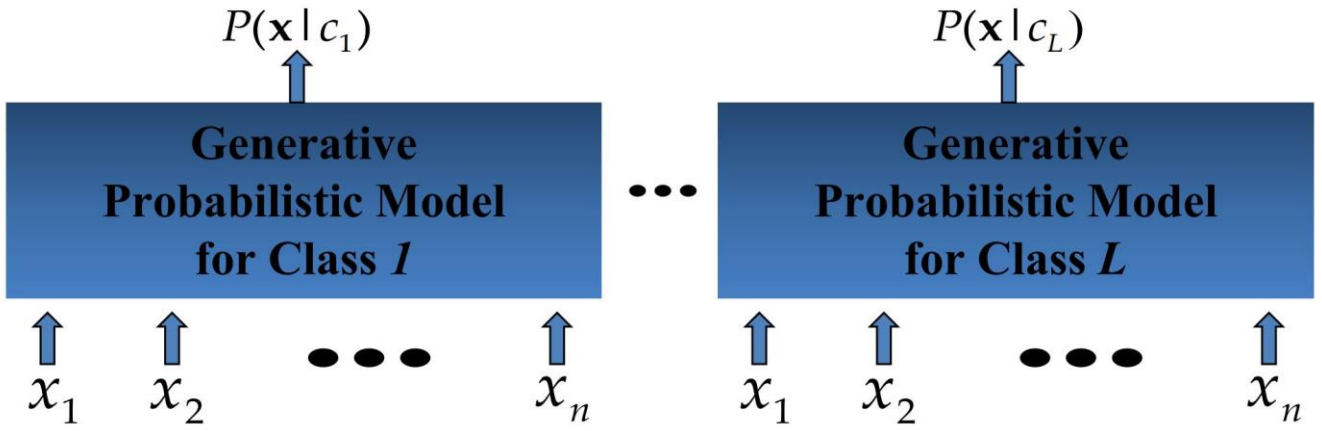
Platt Scaling of SVM outputs

$P(y=1|\mathbf{x})=1/(1+\exp(\mathbf{w}^T\mathbf{x}+b))$

Generative Classifiers

- L probabilistic models have to be trained independently, each is trained on only the examples of the same label, e.g., Hidden Markov Model (HMM), GAN,

$P(\boldsymbol{x}^{(i)}|C_i)$ $P(\mathbf{x}|c) \quad c=c_1,\cdots,c_L, \mathbf{x}=(x_1,\cdots,x_n)$



Discriminative vs. Generative

Bayes Rule

$$\underbrace{p(C \mid X)}_{\text{posterior}} \propto \underbrace{p(X \mid C)}_{\text{likelihood}} \underbrace{p(C)}_{\text{prior}}$$

- Discriminative methods: model posterior
- Generative methods: model likelihood and prior

A Simple Example

- Maximum A Posterior (MAP) Decision:

$$p(zebra \mid image) \text{ vs. } p(no\ zebra \mid image)$$

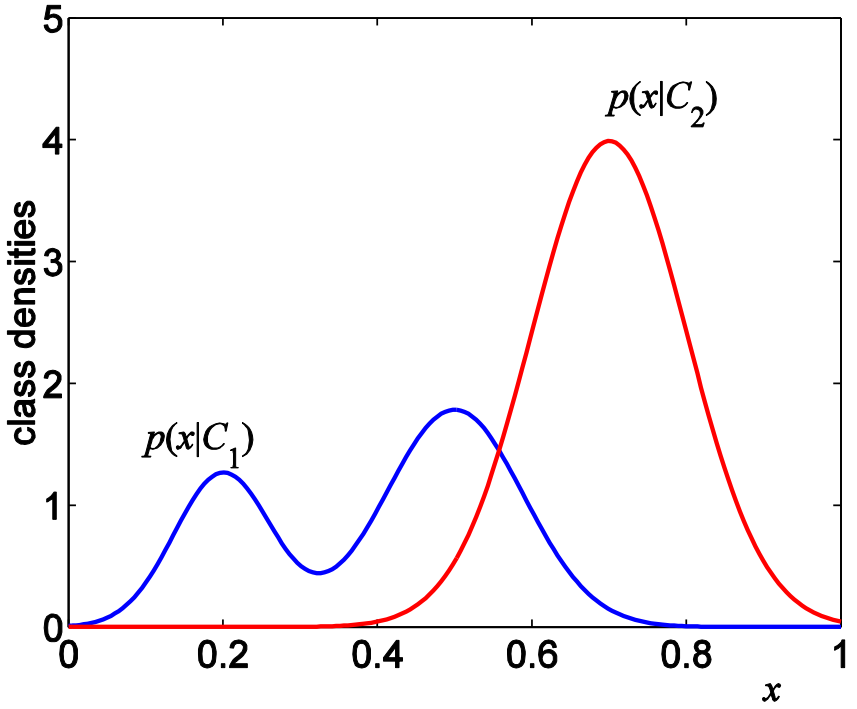
- Bayes Rule

$$p(C \mid X) \propto p(X \mid C)\,p(C)$$

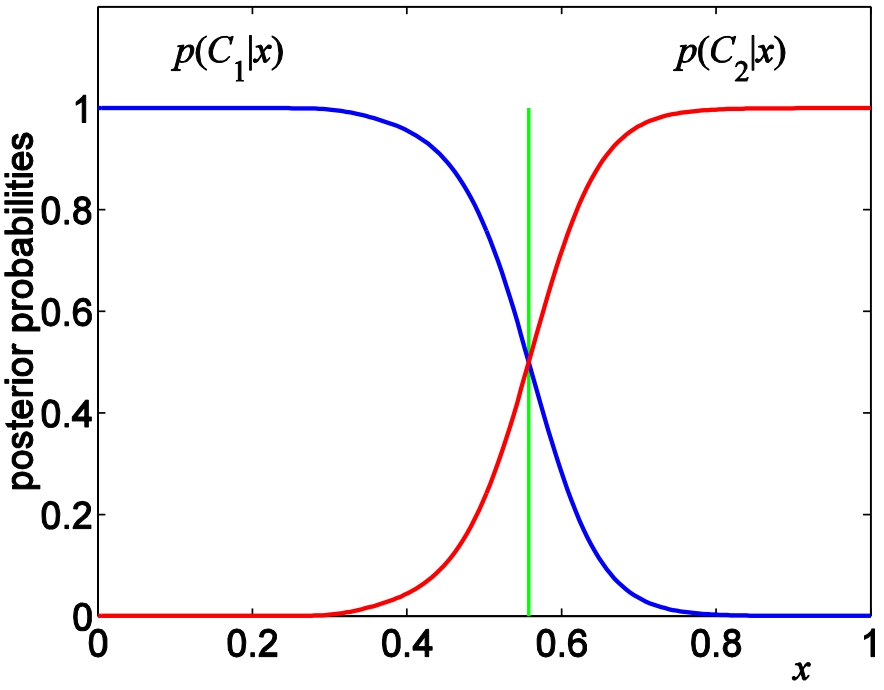
$$\underbrace{p(zebra \mid image)}_{\text{posterior}} \propto \underbrace{p(image \mid zebra)}_{\text{likelihood}}\,\underbrace{p(zebra)}_{\text{prior}}$$

Discriminative vs. Generative

likelihood

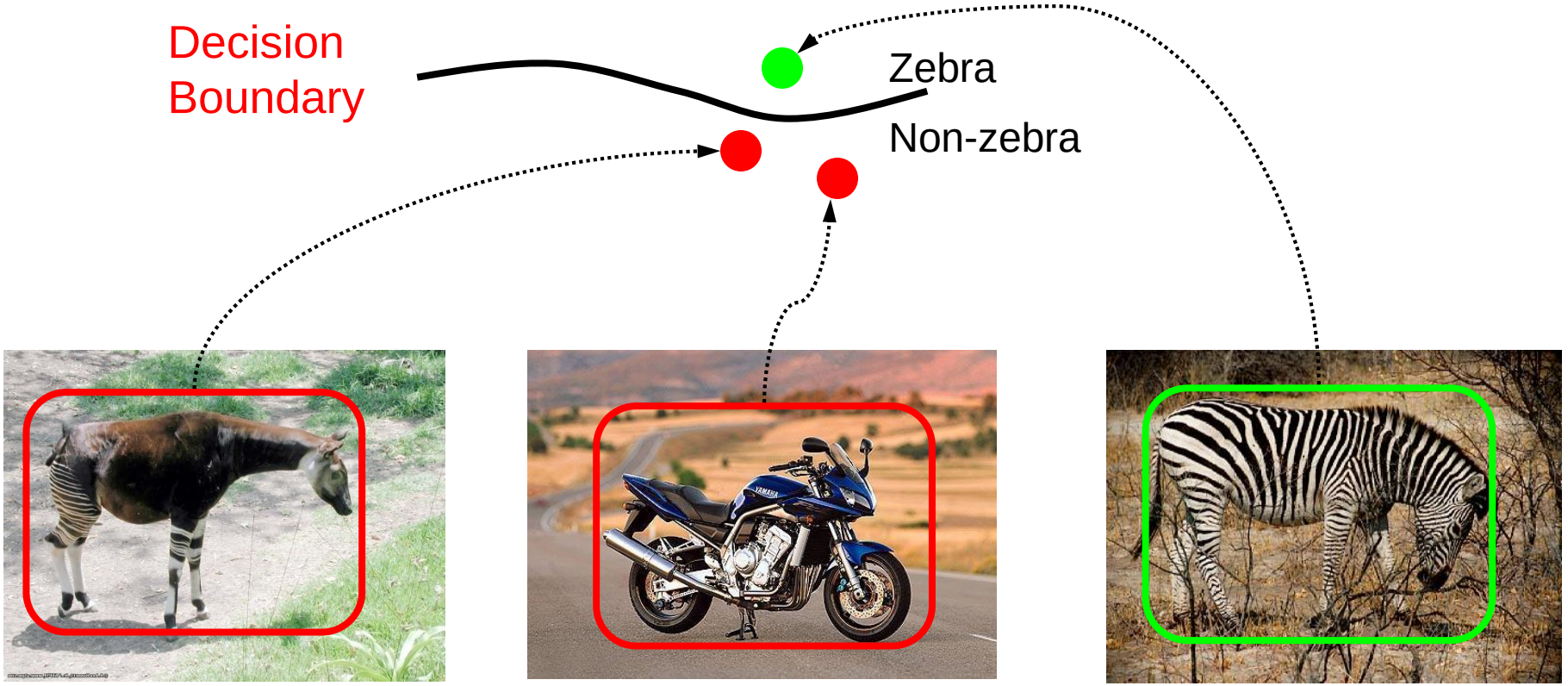


posterior



Discriminative Methods

- Direct modeling of $p(zebra \mid image)$



Generative Methods

- Model $p(image \mid zebra)$ and $p(image \mid no\ zebra)$

$p(image \mid zebra)$	$p(image \mid no\ zebra)$
Low	Middle
High	Middle→Low

Discriminative vs. Generative

- Discriminative methods
  - + Efficient
  - + Often produce better classification rates
  - – Can be hard to interpret (the likelihood and prior are mixed)
  - – Require positive and negative training data
- Generative methods
  - + Interpretable
  - + Can be learned using images from just a single category
  - – Sometimes we don't need to model the likelihood when all we want is to make a decision

Distance Measure of Visual Features

Distance Measures between Two Features

- Suppose two object x and y both have p features

$$x = (x_1, x_2, \cdots, x_p)$$
$$y = (y_1, y_2, \cdots, y_p)$$

- The Minkowski metric is defined by

(r-norm/distance or L^r norm/distance)

$$d(x, y) = \sqrt[r]{\sum_{i=1}^{p} |x_i - y_i|^r}$$

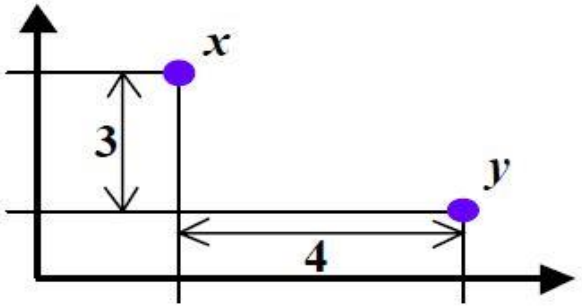
- Most Common Minkowski Metrics

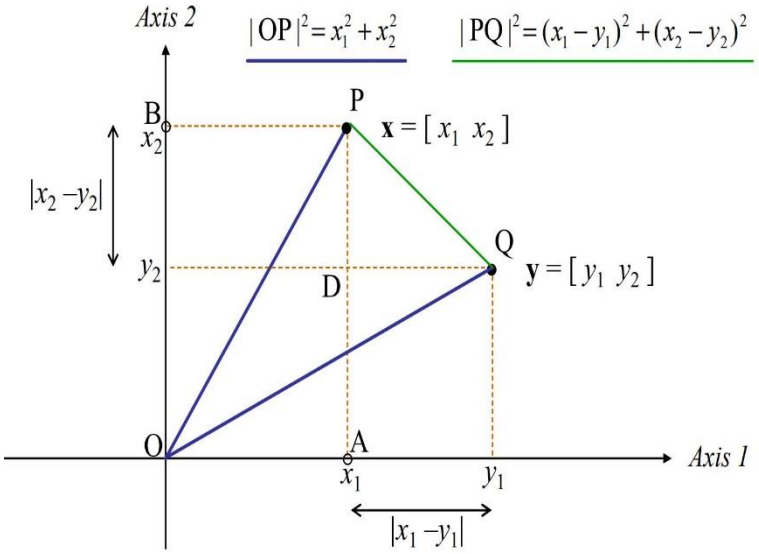
1, $r = 2$ (Euclidean distance) $\quad d(x, y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$

2, $r = 1$ (Manhattan distance) $\quad d(x, y) = \sum_{i=1}^{p} |x_i - y_i|$

3, $r = +\infty$ ("sup" distance) $\quad d(x, y) = \max_{1 \le i \le p} |x_i - y_i|$

Examples of Distance Measures





1 : Euclidean distance : $\sqrt{4^2 + 3^2} = 5.$

2 : Manhattan distance : $4 + 3 = 7.$

3 : "sup" distance : $\max\{4,3\} = 4.$

Manhattan distance is called *Hamming distance* when all features are binary.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
GeneA	0	1	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1
GeneB	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1

Hamming Distance : $\#(01) + \#(10) = 4 + 1 = 5.$

Weighted Euclidean Distance

- When each dimension has quite different mean and standard deviation/variance (dynamic range), e.g.,

$$d(x,y) = \sqrt{(6.0-1.9)^2 + (51-99)^2 + (3.0-2.9)^2} = \sqrt{16.81+2304+0.01} = \sqrt{2320.82}$$
$$= 48.17$$

- Normalized to a standard value of each dimension:

standardized value = (original value – mean) / standard deviation

$$d_{\mathbf{x},\mathbf{y}} = \sqrt{\sum_{j=1}^{J} \left(\frac{x_j}{s_j} - \frac{y_j}{s_j}\right)^2}$$

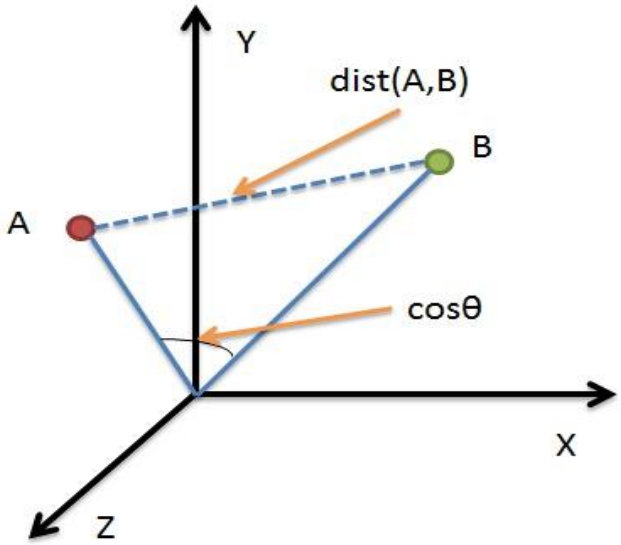
where s_j is the sample standard deviation of the j-th variable.

$$d_{\mathbf{x},\mathbf{y}} = \sqrt{\sum_{j=1}^{J} \frac{1}{s_j^2}(x_j - y_j)^2}$$

where $w_j = 1/s_j^2$ is the inverse of the j-th variance.

$$= \sqrt{\sum_{j=1}^{J} w_j (x_j - y_j)^2}$$

Cosine Distance/Similarity



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

cosine-distance = 1-(cosine-similarity), [0,2]

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

Histogram Dissimilarity

- Two (normalized) histograms: multivariate distribution
  - Color information, HOG, LBP, etc

$$X = [x_1, x_2, \ldots, x_p]^T, \quad |X| = \sum_{i=1}^{p} x_p = 1$$

$$Y = [y_1, y_2, \ldots, y_p]^T, \quad |Y| = \sum_{i=1}^{p} y_p = 1$$

  - Bin-by-bin Dissimilarity (distance) measure
    - Only compare the corresponding histogram bins
    - i.e., only compare x_i and y_j, if $i=j$
  - Cross-bin Dissimilarity (distance) measure (earth mover distance)
    - Compare non-corresponding histogram bins
    - Need to define the distance between different bins

Some Bin-by-Bin Distances

- Minkowski Distance

$$d_{L_r}(X,Y) = \left(\sum_i | x_i - y_i |^r \right)^{1/r}$$

- Histogram Intersection

$$d_{\cap}(X,Y) = 1 - \sum_i \min(x_i, y_i)$$

- Kullback-Leibler Divergence (asymmetric)

$$d_{KL}(X,Y) = \sum_i x_i \log \frac{x_i}{y_i} \geq 0$$

- Jeffrey Divergence (symmetric)

$$d_J(X,Y) = \sum_i (x_i \log \frac{x_i}{m_i} + y_i \log \frac{y_i}{m_i}), \quad m_i = \frac{x_i + y_i}{2}$$

- Cross Entropy $D_{CRO}(\mathrm{X}, \mathrm{Y}) = -\sum_i x_i \log y_i$