Machine Learning for Satellite Image Analysis

Prof. Krishna Mohan Buddhiraju

IIT Bombay

bkmohan@csre.iitb.ac.in

Lecture delivered on 04 March 2024 for NRSC visitors

Historical Introduction

- ***Machine Learning*** – coined by Arthur Samuel (IBM) in the 1952
- Computer was programmed to play Checkers (board game)
- The computer played better by playing with human players played ➔ computer was *observing* the way human players played the game
- In other words, the computer made better moves by *learning* the game from observing the way human players played

Historical Introduction contd…

- ***Perceptron –*** Proposed by Frank Rosenblatt in 1957 at Stanford University
- ***Delta rule –*** proposed by Paul Werbos to train the Perceptron
- ***Perceptron –*** Used in applications
- ***Minsky and Papert (MIT)*** – Proved that the Perceptron cannot handle cases where the data are not linearly separable
- ***Multilayer Perceptrons –*** thought to be a solution to the problem, however a viable training algorithm was not available
- ***Rumelhart et al.*** (MIT, 1986) proposed the backpropagation algorithm to train the multilayer perceptron network!!!

Historical Introduction contd…

- ***Expert Systems –*** Rule based, application specific (prominent among them, the work at Stanford University, later at SRI International)
- ***Hierarchical Decision Trees –*** ID3, CART, …
- ***Natural Language Processing***
- ***Support Vector Machine***
- ***Adaboost –*** Strong classifier from an ensemble of weak classifiers

Historical Introduction contd…

- ***Self-driving car (1994) –*** first tested
- ***Deep Blue (IBM supercomputer) –*** defeated Garry Kasparov in chess in 1997
- ***Self-driving car (2009)*** – introduced by Google
- ***Watson (IBM supercomputer, 2011)*** – won the game of Jeopardy against humans
- ***Human vision*** (in a limited sense) – matched by Machines in 2014
- ***Machine Translation (driven by neural networks) –*** became very successful in 2015

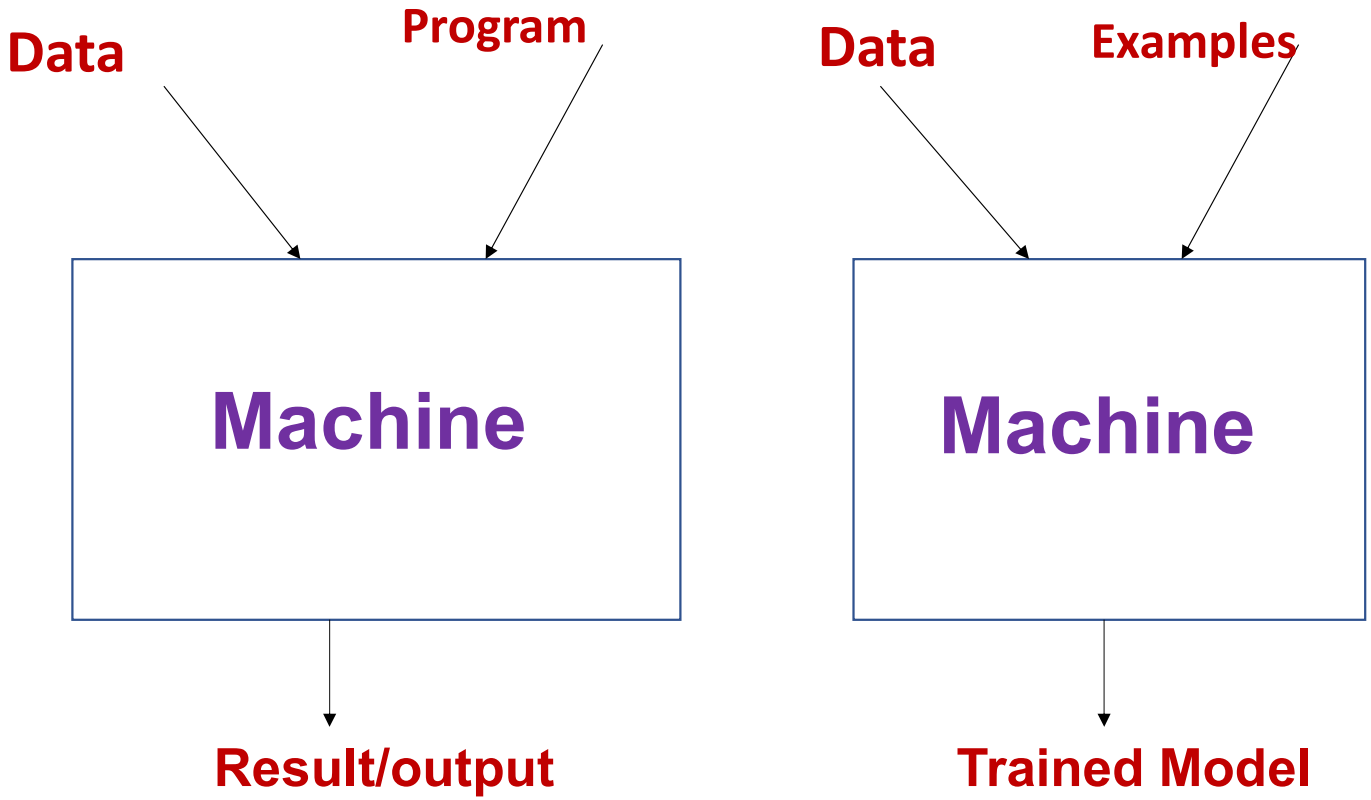
Historical Introduction contd…

- ***Advances in Hardware –*** CPU clusters and Graphical Processing Units (GPU)'s revolutionized machine learning applications
- ***Paradigms –*** Cloud computing, Big Data
- ***Frameworks – TensorFlow, Keras, …***

What is Learning? What is a Learning Algorithm?

Programmed v/s Learning Systems

Source: Prof. Sudheshna Sarkar – NPTEL course on Machine Learning



Formal Definition of Learning

Ability to improve one's behaviour with experience

Change in the state of a system by which it can perform the same task more efficiently next time

"**A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.**"

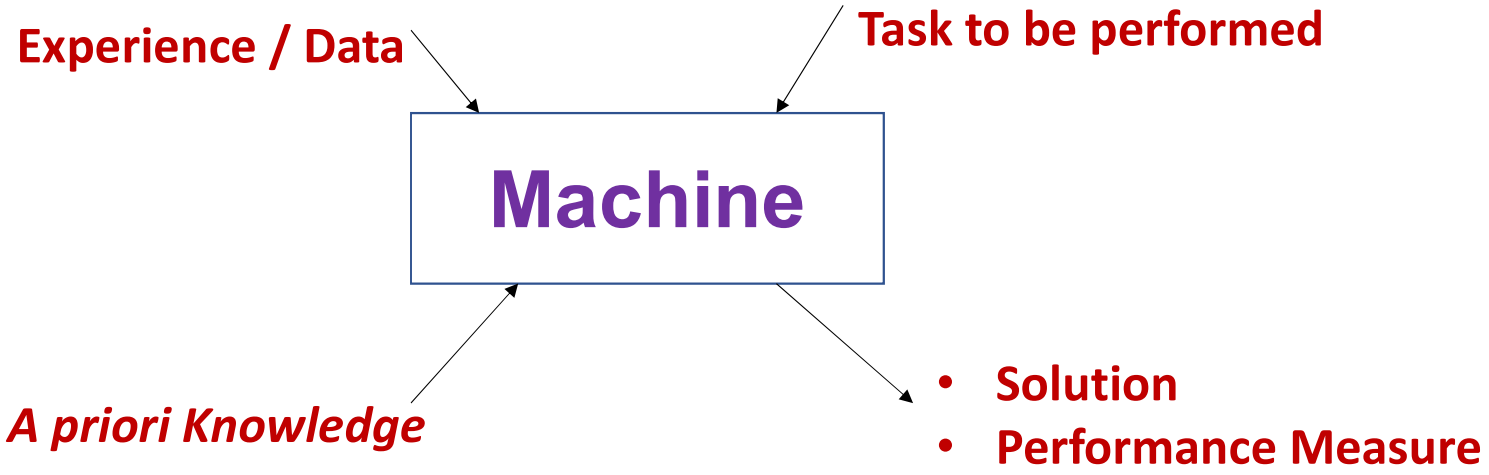
(Modern Definition by ML expert Prof. Tom Mitchell)

Examples

- Medical diagnosis
- Driving a car
- Robot navigation
- Identifying laduse/landcover classes of pixels given a satellite image
- Speech recognition and machine translation
- Online advertising (will a browsing person click on an advertisement?)
- Forecasting stockmarket movement, currency rate movement

What do we wish to achieve by machine learning?

- ***Develop algorithms to:***
  - Learn from data
  - Build models relating the ***features*** that characterize the data to the ***knowledge*** intended to be derived from data
  - Deploy models to perform useful tasks
- ***Based on Tom Mitchell's definition:***



How is Machine Learning Used?

- ***Learning***
  - Combine prior knowledge, data
  - Build models
- ***Reasoning/Generalization***
  - Use the models to solve a new problem
  - Produce solution with measurable performance

Building a Machine Learning System

- *Choose data (features, examples)*
- *Choose target function (discriminant function)* ← *simple/very complex*
- *Choose type of target function (linear / nonlinear discriminant)*
- *Algorithm to find the target function*

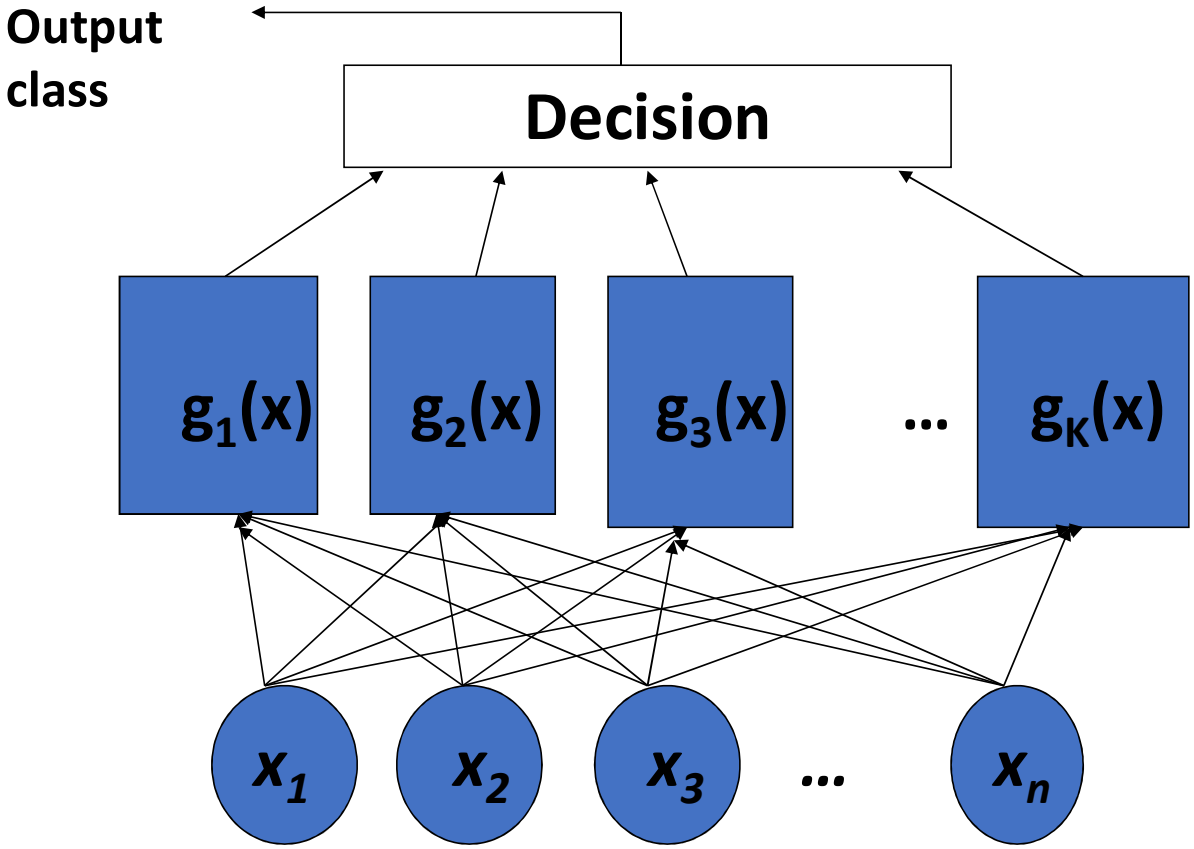
Discriminant Function Point of View of Classification

- A discriminant function $g_k(X)$ assumes high value when a sample X belongs to class k

  - Set of discriminant functions $g_i(x)$, $i = 1,\ldots, c$

  - The classifier assigns a feature vector x to class ω_k if:

$$g_k(x) > g_j(x) \quad \forall j \neq i$$


Output class
Decision
$g_1(x)$
$g_2(x)$
$g_3(x)$
...
$g_K(x)$
x_1
x_2
x_3
...
x_n

Features

- Features are attributes of the data elements based on which the elements are assigned to various classes.
- E.g., in satellite remote sensing, the features are measurements made by sensors in different wavelengths of the eLecture romagnetic spectrum – visible/ infrared / microwave/texture features …

Features

- In medical diagnosis, the features may be the temperature, blood pressure, lipid profile, blood sugar, and a variety of other data collected through pathological investigations
- The features may be qualitative (high, moderate, low) and/or quantitative.
- The classification may be presence of heart disease (positive) or absence of heart disease (negative)

Supervised Learning

- The classifier has the advantage of an analyst or domain knowledge using which the classifier can be guided to learn the relationship between the data and the classes.
- The number of classes, prototype pixels for each class can be identified using this prior knowledge
- Given (X_i, y_i) ← X_i are the feature vectors
  y_i are the labels

Unsupervised Learning

- When access to domain knowledge or the experience of an analyst is missing, the data can still be analyzed by numerical exploration, whereby the data are grouped into subsets or ***clusters*** based on statistical similarity
- Given X_i the feature vectors but no class labels

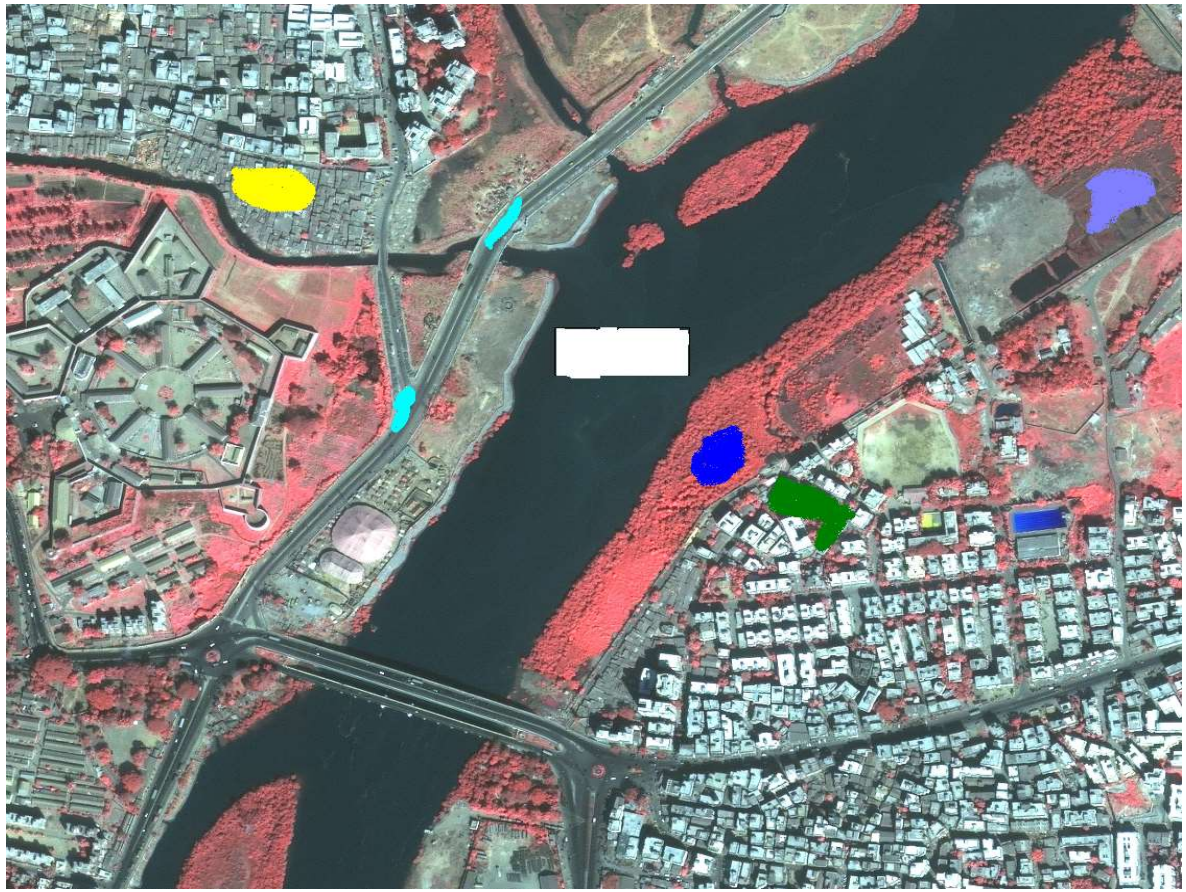
Example from Satellite Imaging

Prototype Pixels for Different Classes

- The prototype pixels are ***samples*** of the population of pixels belonging to each class
- The size and distribution of samples are formally governed by the mathematical theory of sampling
- There are several criteria for choosing the samples belonging to different classes

Landuse/Land-cover classes:

- Water
- Mangroves
- Roads
- Buildings
- Slum areas
- Salt Pan areas



Features:

Reflectance in

- Blue
- Green
- Red
- Near-Infrared

wavelengths

What does the classifier *compute from training samples?*

- **Typical characteristics of classes**
  - Mean vector
  - Covariance matrix
  - Minimum and maximum gray levels within each band
  - Conditional probability density function $p(\mathbf{x} \mid C_i)$ where C_i is the i^{th} class and $\mathbf{x}$ is the feature vector
- **What does the classifier *learn from the samples?***
  - Class boundaries separating the classes
  - Hyperplanes in linear case or more complex surfaces in nonlinear case

Sample Programmed Classifier

Maximum Likelihood Classifier

- Calculates the likelihood of a pixel being in different classes conditional on the available features, and assigns the pixel to the class with the highest likelihood

Likelihood Calculation

$P(C_i|\mathbf{x})$ is computed using Bayes' Theorem in terms of $P(\mathbf{x}|C_i)$

$P(C_i|\mathbf{x}) = P(\mathbf{x}|C_i)\, P(C_i) / P(\mathbf{x})$

$\mathbf{x}$ is assigned to class C_j such that

$P(C_j|\mathbf{x}) = \text{Max}_i\, P(C_i|\mathbf{x})$, i=1…K, the number of classes.

$P(C_i)$ is the prior probability of occurrence of class i in the image

$P(\mathbf{x})$ is the multivariate probability density function of feature $\mathbf{x}$.

Likelihood Calculation

- P($\mathbf{x}|C_i$) is assumed to be multivariate Gaussian distributed in practical parametric classifiers.
- Gaussian distribution is mathematically simple to handle.
- Each class conditional density function P($\mathbf{x}|C_i$) is represented by its mean vector $\boldsymbol{\mu}_i$ and covariance matrix Σ_i

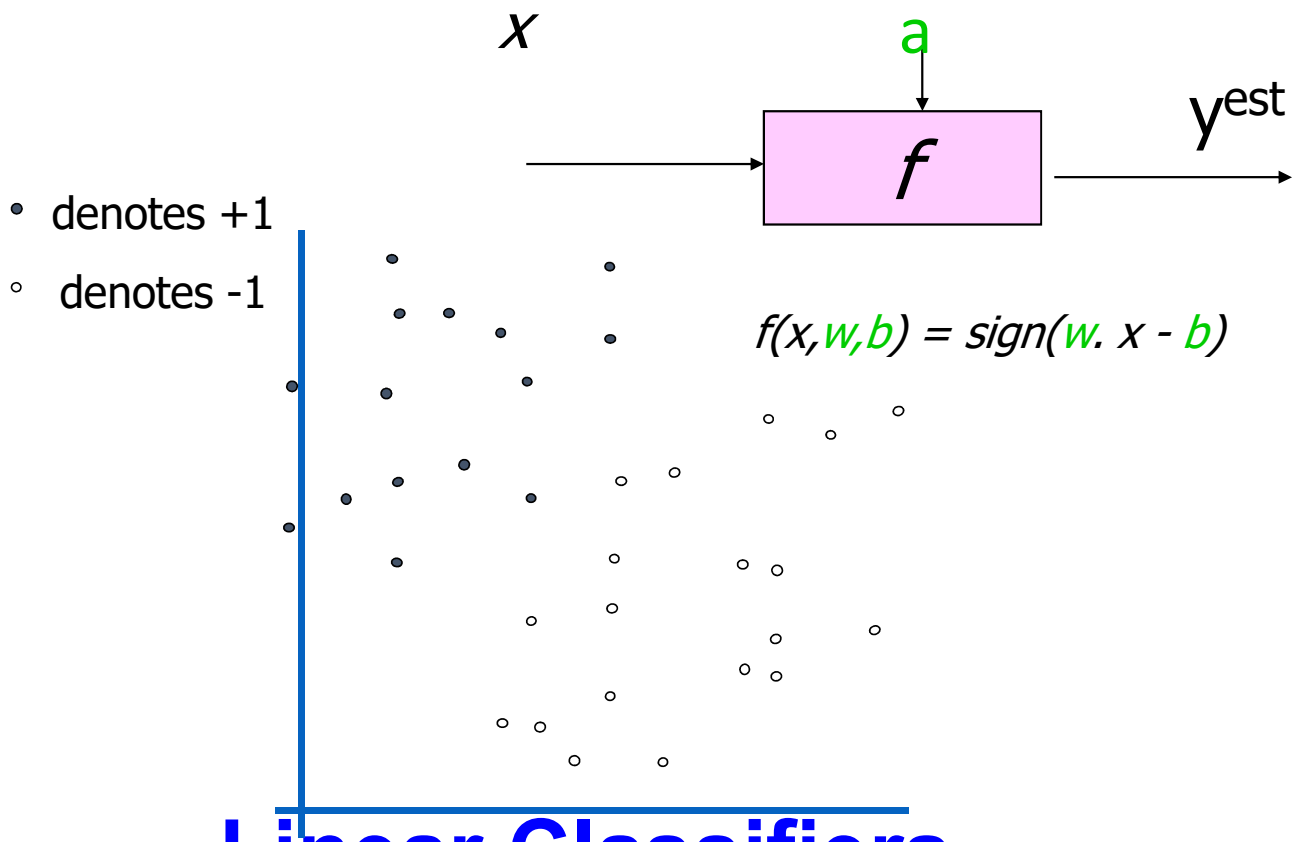
$$p(\mathbf{x} \mid C_i) = \frac{1}{(2\pi)^{L/2} \mid \Sigma_i \mid^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathbf{T}} \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)}$$

Likelihood Calculation

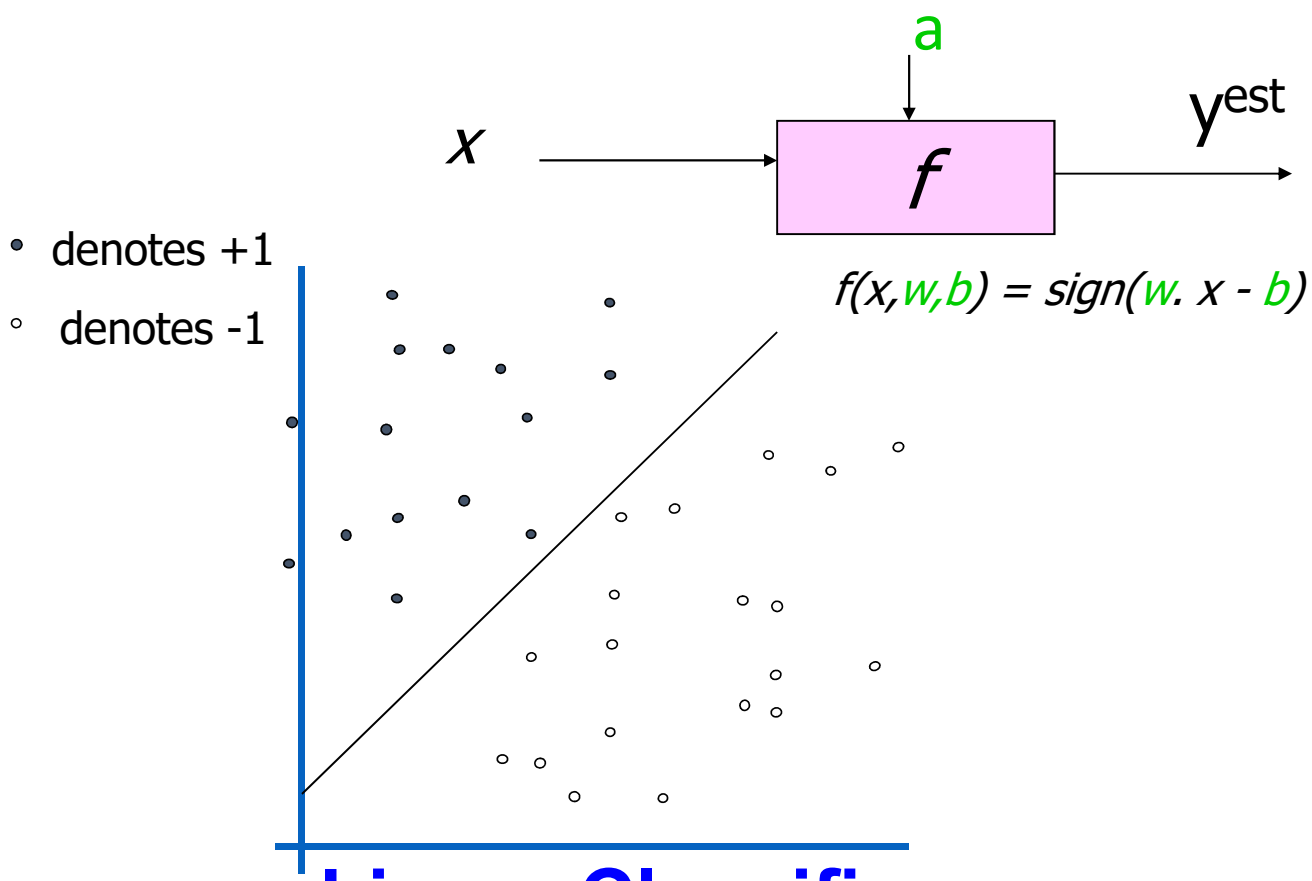
$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) - \frac{L}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\mathrm{C}_i)$$

- We assume that the covariance matrices for each class are different.
- The term $(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)$ is known as the Mahalanobis distance between x and μ_i (after Prof. P.C. Mahalanobis, famous Indian statistician and founder of Indian Statistical Institute)

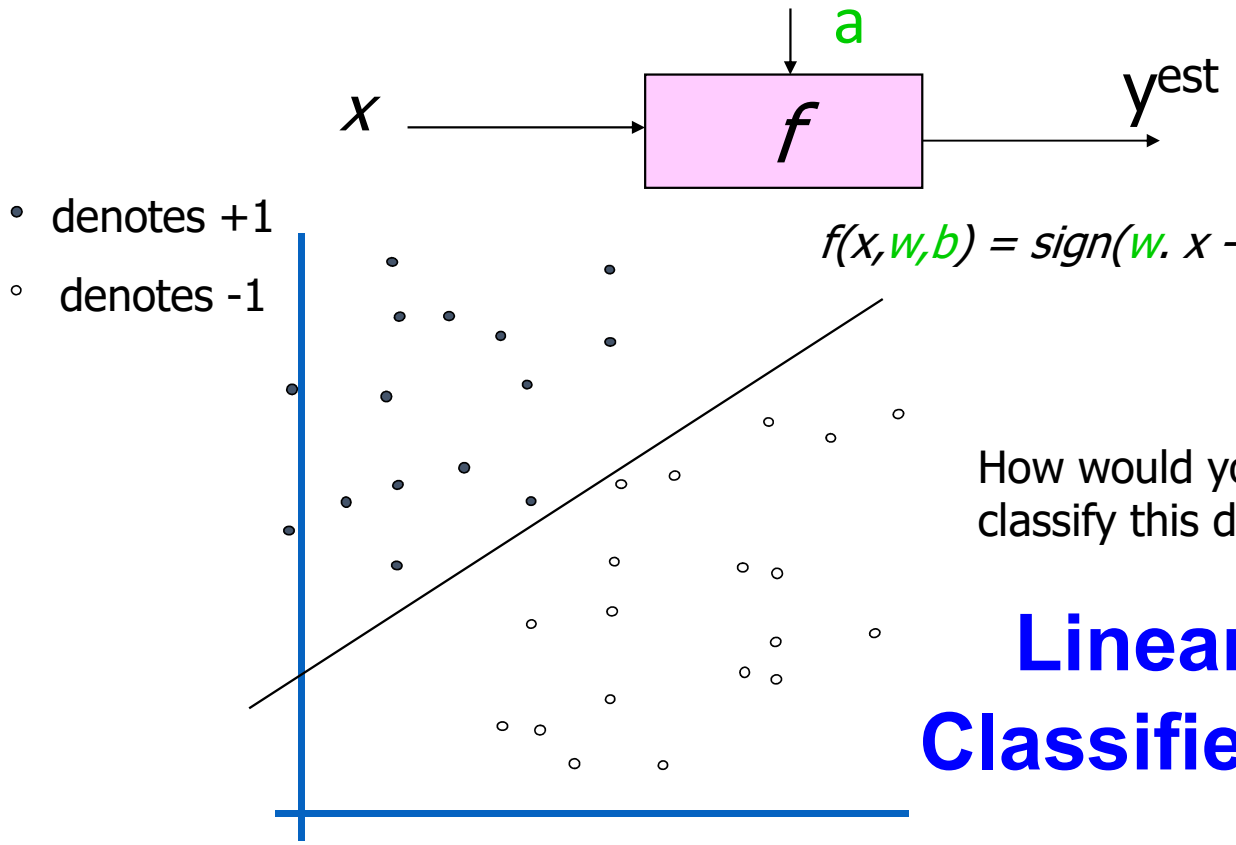
Learning Based Classifier
Support Vector Machine



Linear Classifiers

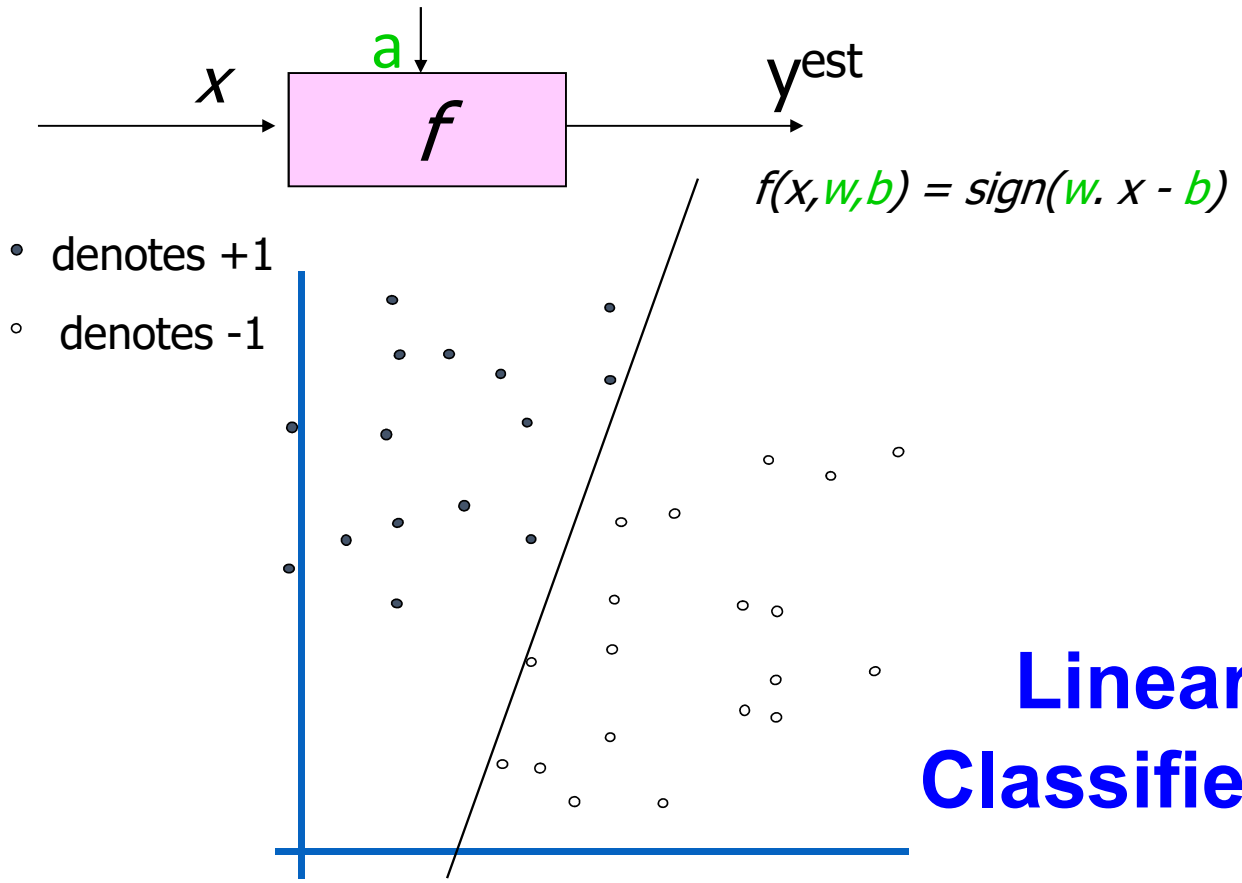


Linear Classifiers

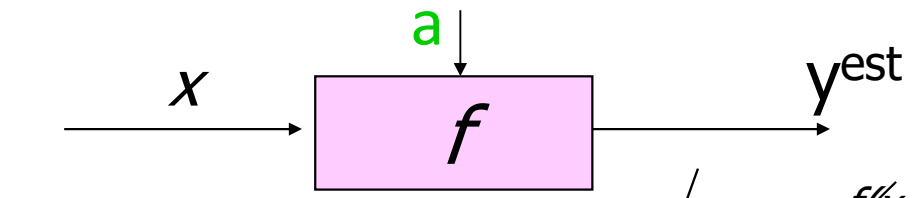


How would you classify this data?

Linear Classifiers

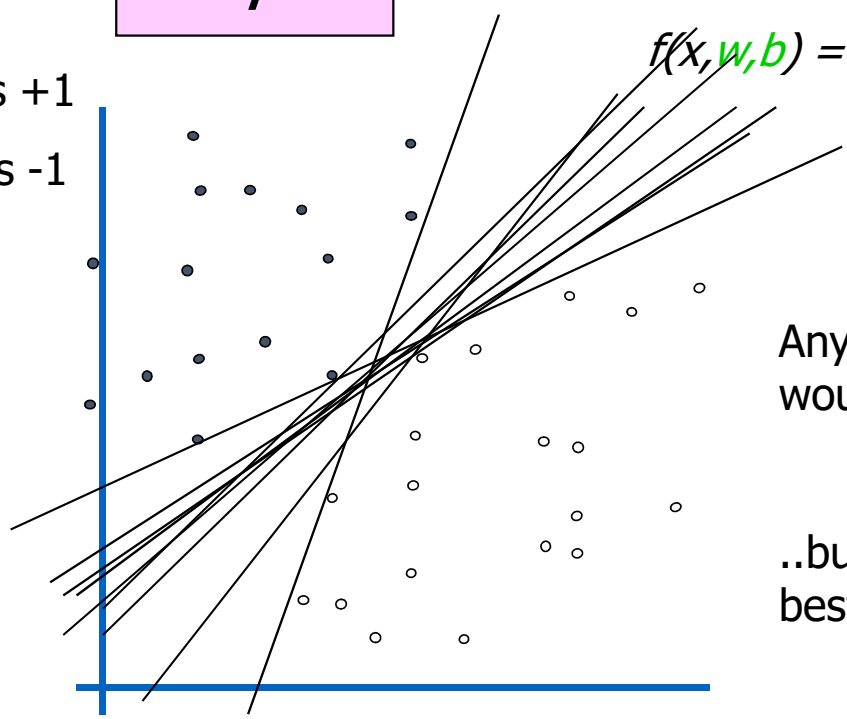


Linear Classifiers



$f(x,w,b) = sign(w.\ x - b)$

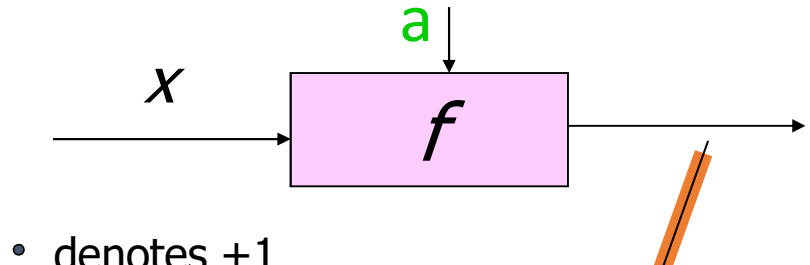
- denotes +1
- denotes -1



Any of these would be fine..

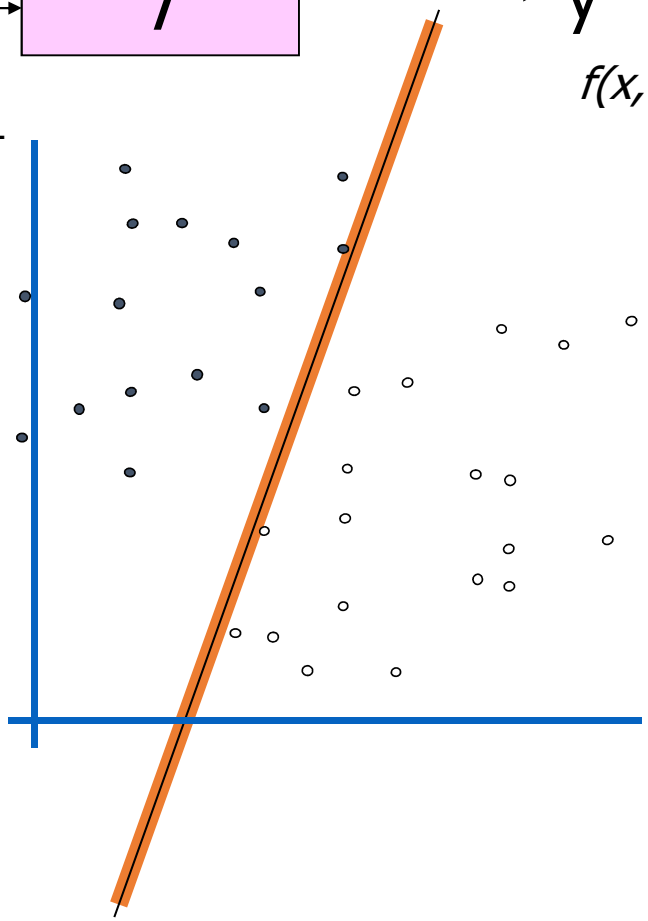
..but which is best?

Margin



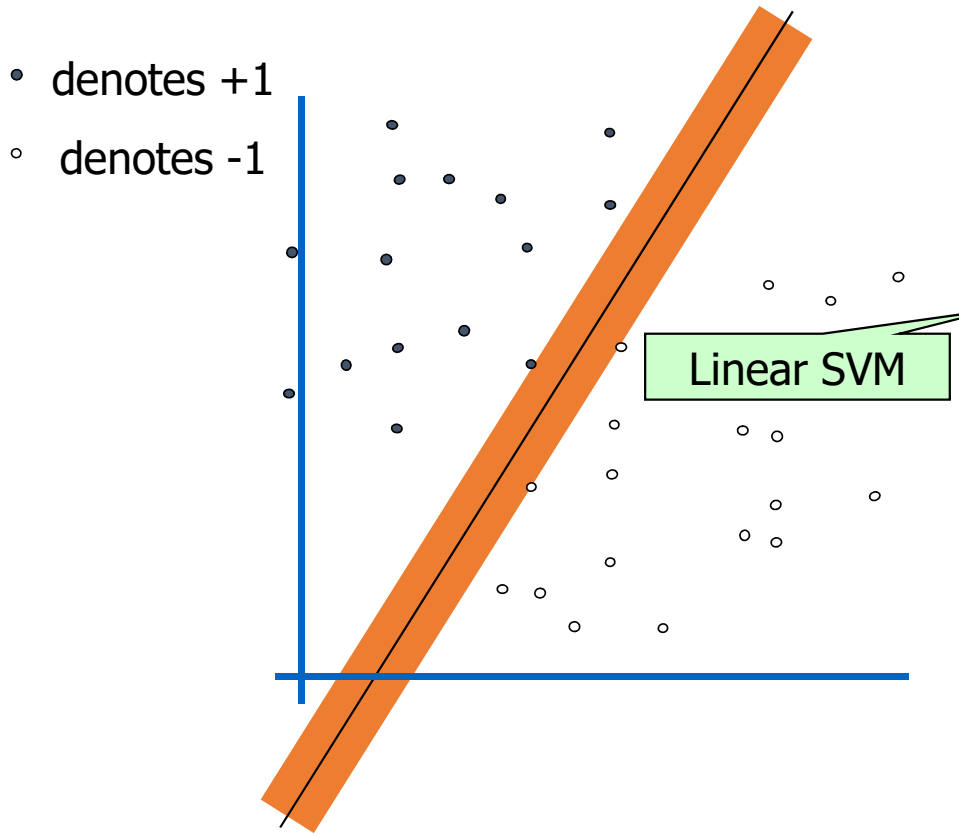
$f(x,w,b) = sign(w.\ x - b)$

- denotes +1
- denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

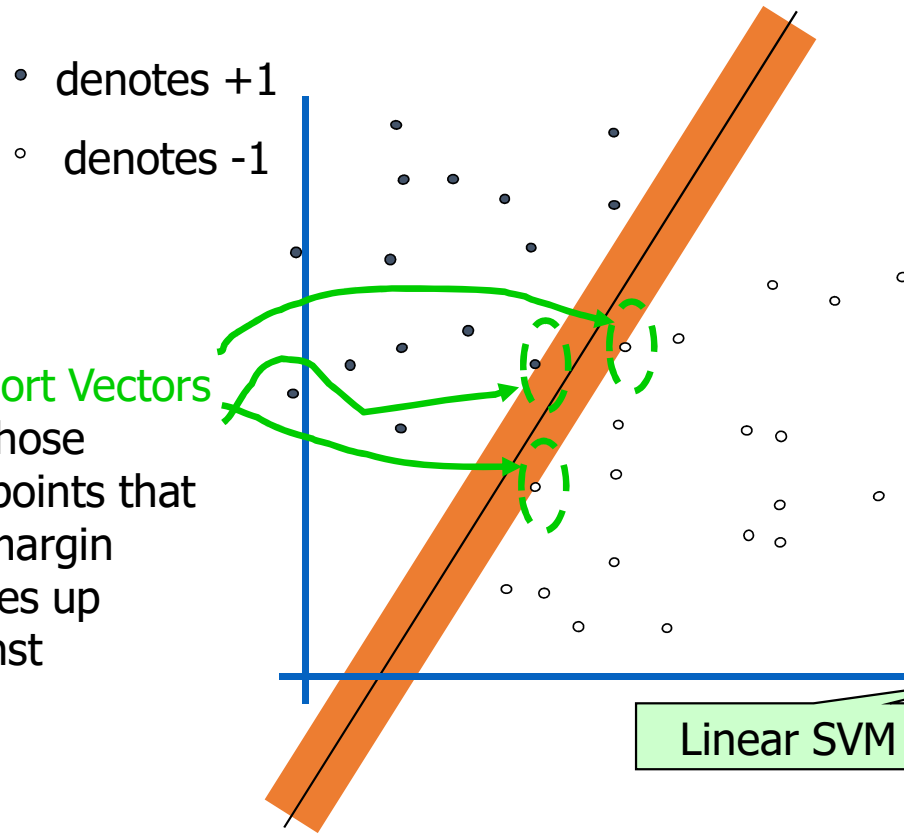
Maximum Margin



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

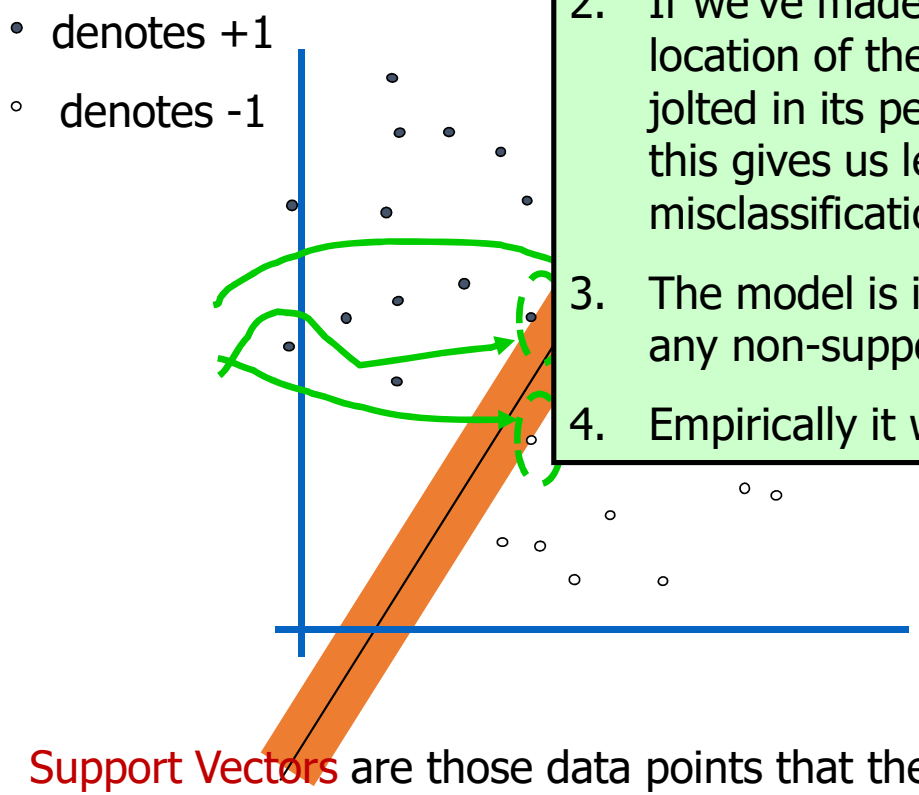
Maximum Margin



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

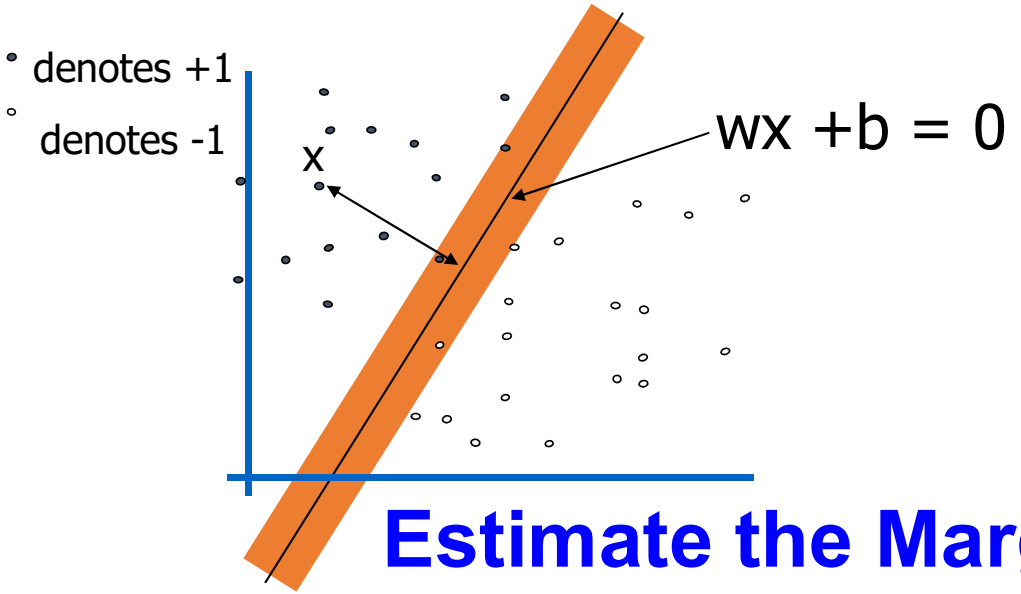
Why Maximum Margin?



1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. The model is immune to removal of any non-support-vector datapoints.
4. Empirically it works very very well.

This is the simplest kind of SVM (Called an LSVM)

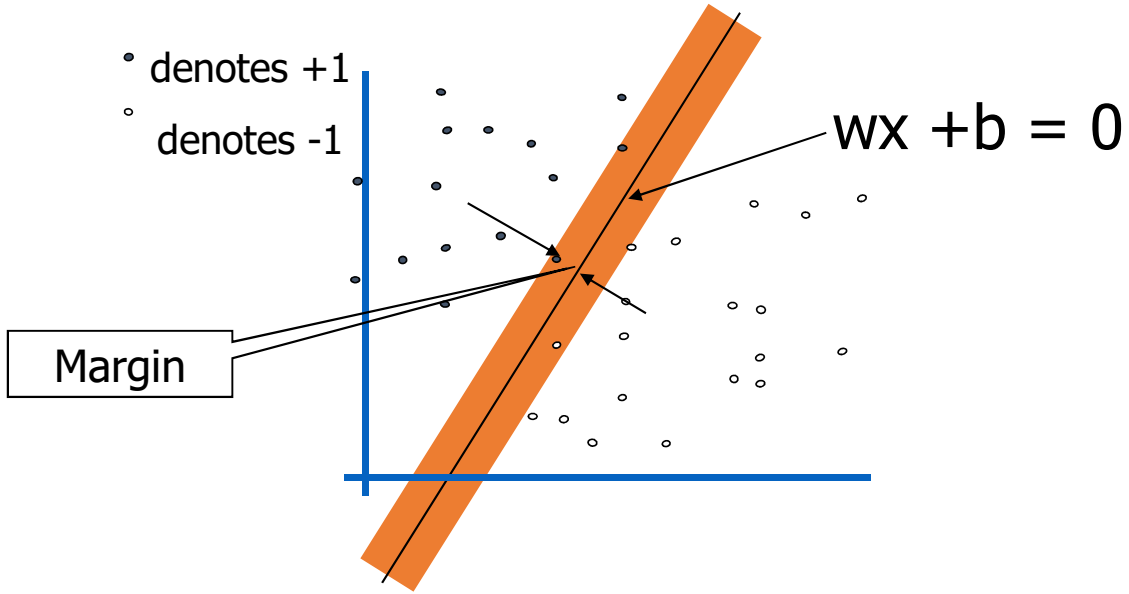
Support Vectors are those data points that the margin pushes up against



Estimate the Margin

- What is the distance expression for a point **x** to a line **wx**+b= 0?

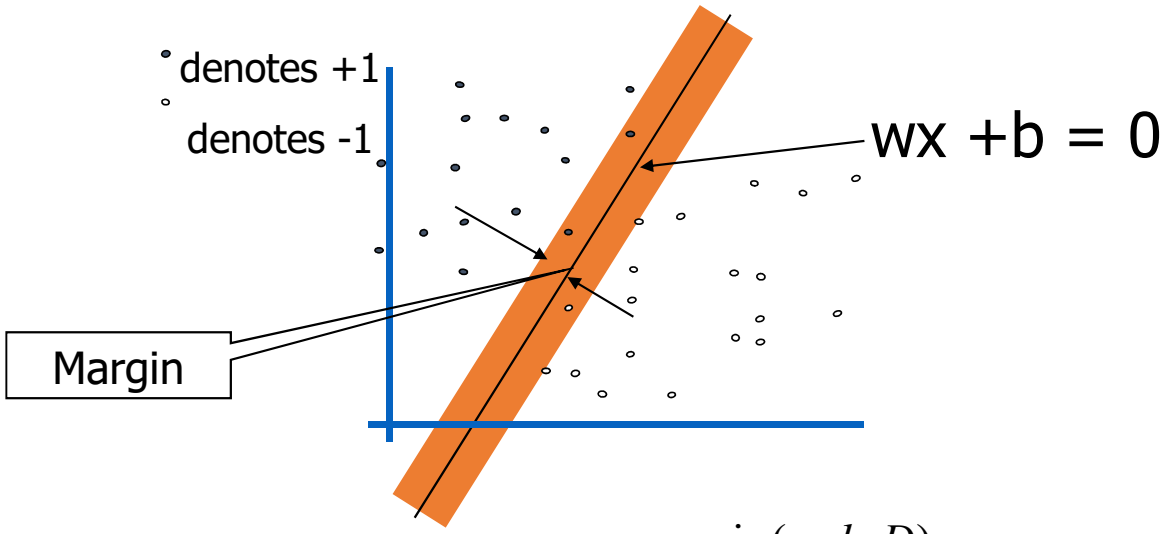
$$d(\mathbf{x}) = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$



- What is the expression for margin?

$$\text{margin} \equiv \arg\min_{\mathbf{x}\in D} d(\mathbf{x}) = \arg\min_{\mathbf{x}\in D} \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Maximize Margin

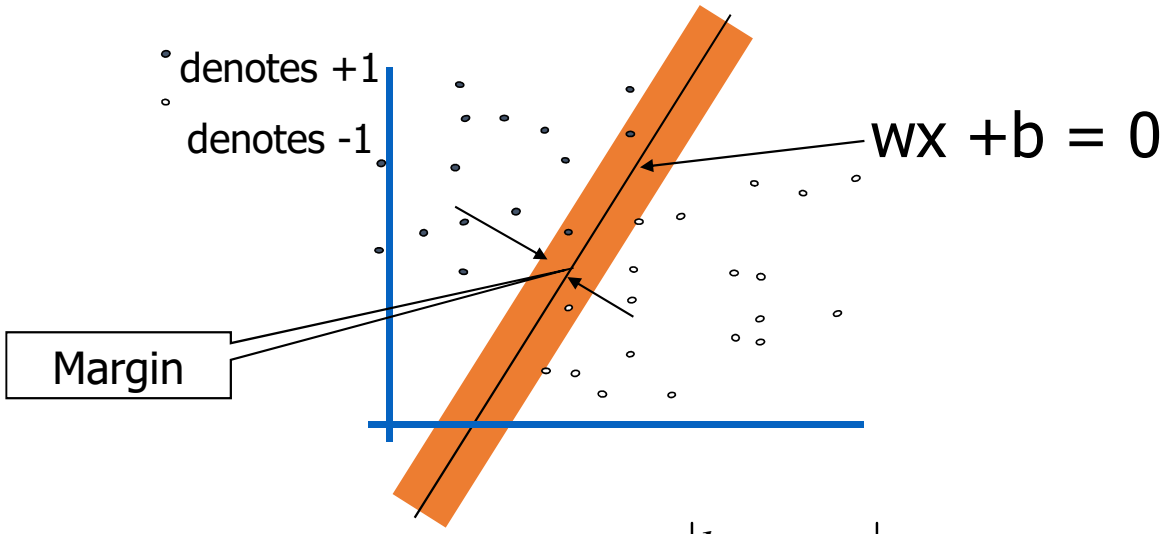


$$\underset{\mathbf{w},b}{\text{argmax}}\ \text{margin}(\mathbf{w},b,D)$$

$$= \underset{\mathbf{w},b}{\text{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min}\, d(\mathbf{x}_i)$$

$$= \underset{\mathbf{w},b}{\text{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

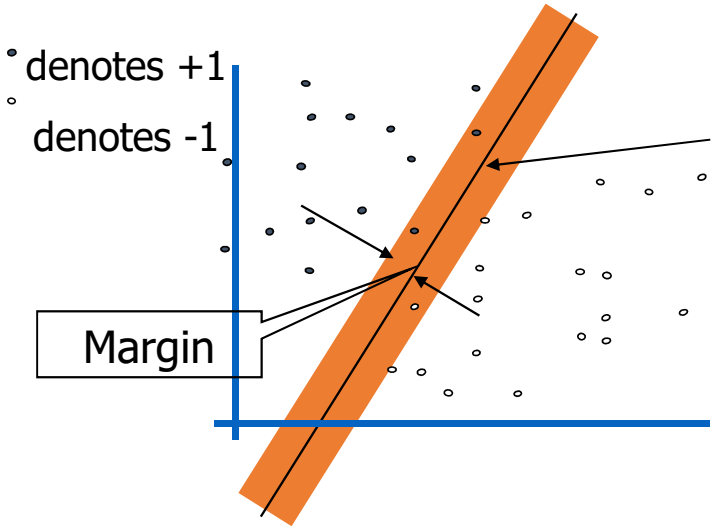
Maximize Margin



$$\underset{\mathbf{w},b}{\operatorname{argmax}} \; \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) > 0$$

Maximize Margin



$$\underset{\mathbf{w},b}{\operatorname{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$$

Strategy:

$$\forall \mathbf{x}_i \in D : \left|b + \mathbf{x}_i \cdot \mathbf{w}\right| \geq 1$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$$

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\text{argmax}} \sum_{k=1}^{d} w_k^2$$

subject to

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1$$

....

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1$$

Maximum Margin Linear Classifier

There are challenges in coming up with the ideal optimization solution

- This is a standard constrained quadratic optimization problem solvable using known techniques, described in detail in most standard textbooks

Classification Example

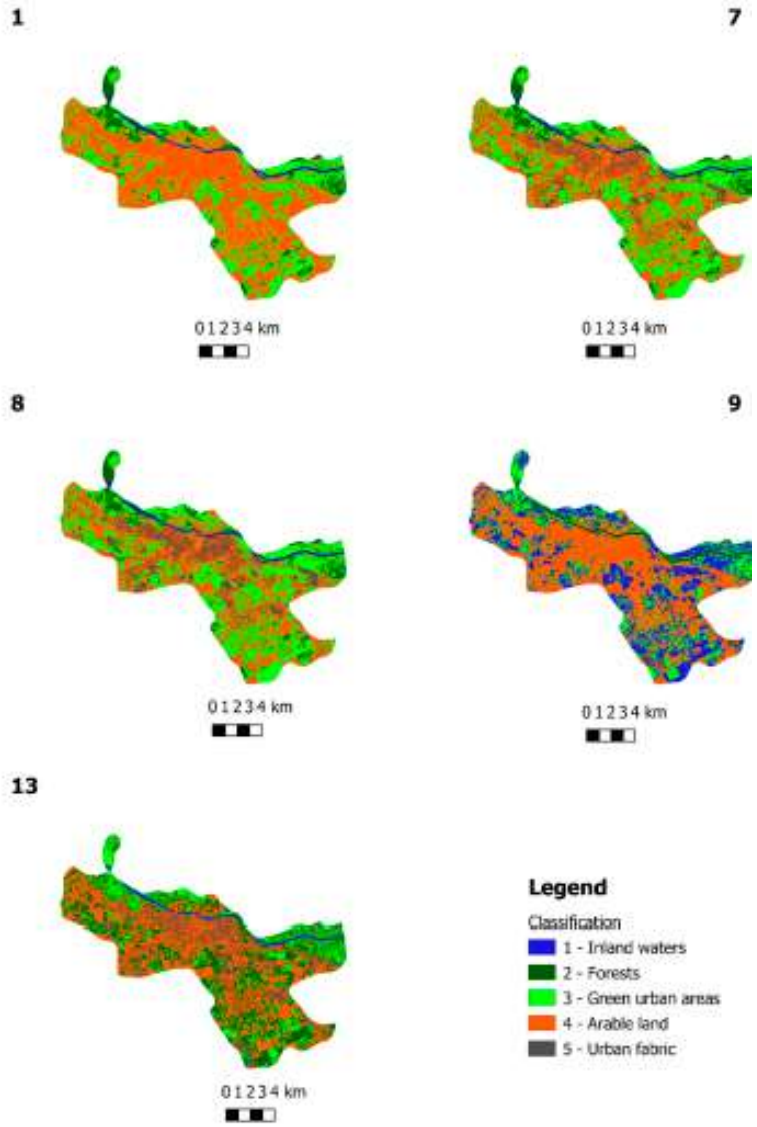
Source: MDPI

Journals
Remote Sensing
Volume 11
Issue 6
10.3390/rs11060655

SVM Classification Using Different Nonlinear Kernels

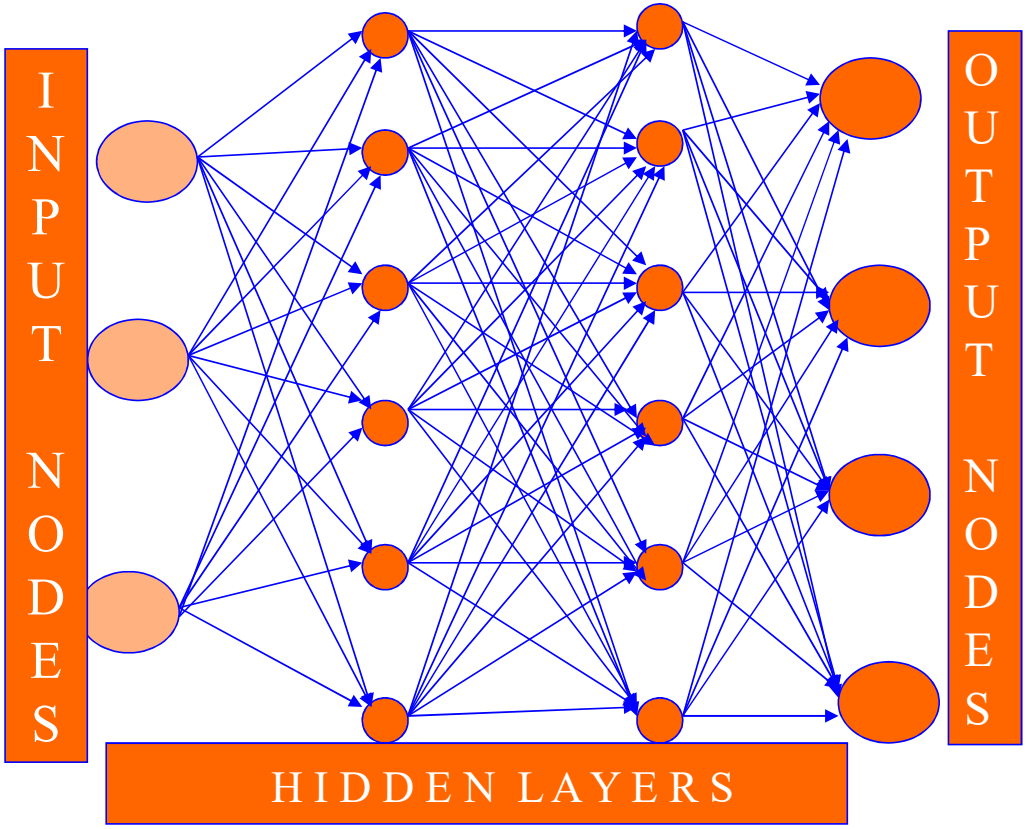
Source: MDPI

Journals
Remote Sensing
Volume 11
Issue 6
10.3390/rs11060655



Another Learned Classifier
Multilayer Perceptron

Multilayer Perceptron Network



Network Architecture

- Input layer – all the features are fed to the classifier from the input nodes
- Classification output is observed at the output layer
- Between input and output layers there are ***hidden*** layers that are computation layers
- End users do not know the computations happening in the hidden layers

Problem to be solved

- Train the classifier to separate samples into their correct classes
- Requires *learning* of correct link weights connecting nodes of one layer to the nodes of the subsequent layer
- Knowledge relating the input features to the desired classes lies in the link weights
- Strategy is to randomly initialize the weights, determine the classification error and update weights as a function of the negative of error gradient)

Training the Multilayer Perceptron Network

- Total Error computed per training sample over all neurons in the output layer

$$\xi(n) = \frac{1}{2}\sum_{j \in C} e^2(n)$$

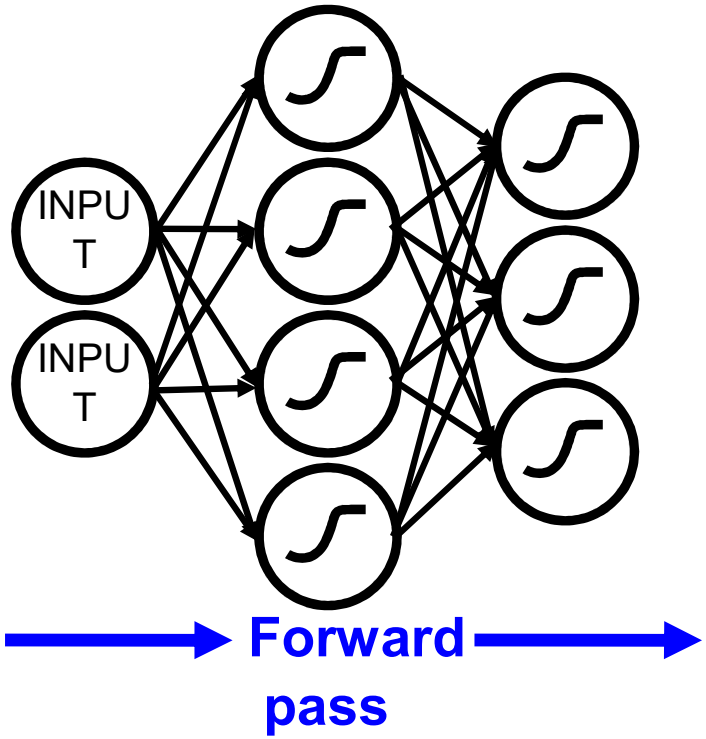
Used in online learning

- If there are N training samples, then the average error can be computed as

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N} \xi(n)$$

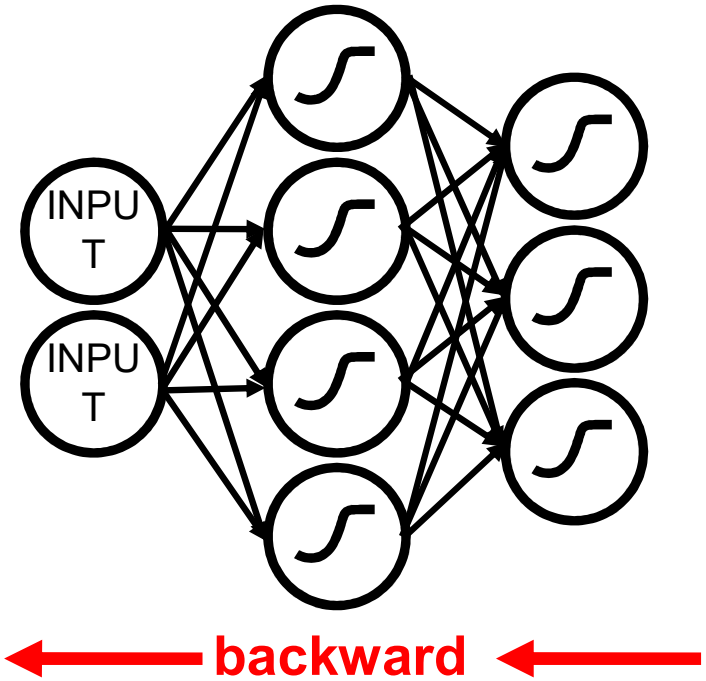
- **Used in batch learning**

Multilayer Feed-forward Network



Signals flow from the input layer towards the output layer

Error is backpropagated!

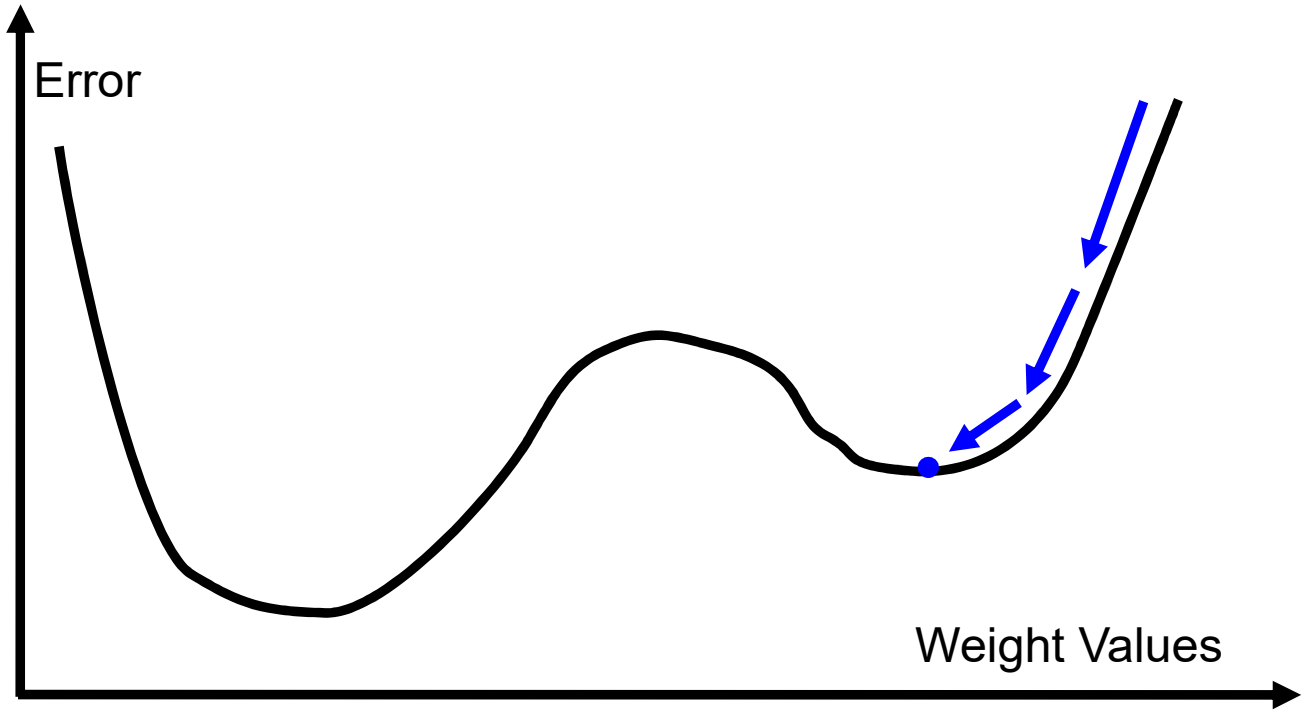


Error signal flows from output layer backwards towards the input layer, updating the weights along the way

Error Backpropagation

- The algorithm takes its name – backpropagation – from the fact that during training, the error is propagated back, from the output to the hidden layer!

Gradient descent based error minimization



Selected Applications of MLP Classifier

- Landuse/Landcover classification
- Edge and line detection

Supervised Image Classification

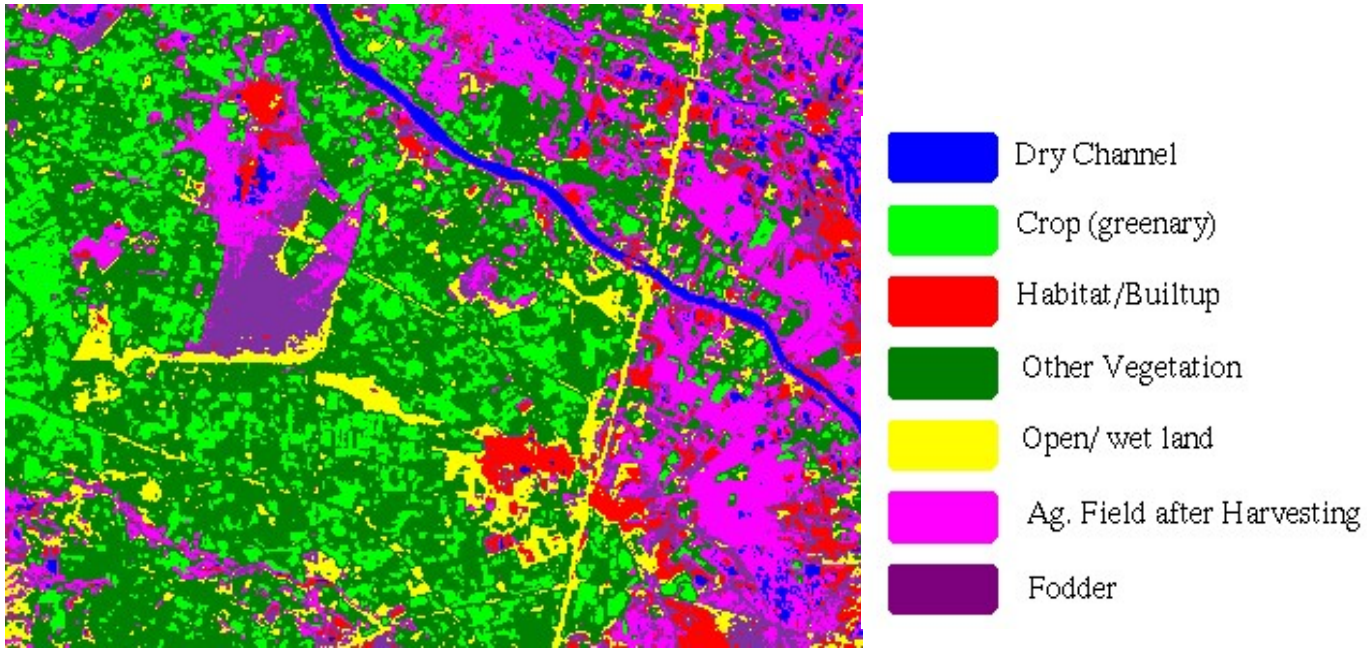
- Identify the number of classes
- Identify the training data and generate the training patterns
- Define the network
  - Input/Output layers
  - Hidden layers
  - Gain and momentum terms

Supervised Image Classification

- Size of input layer = number of bands in the input data
- Size of output layer = number of classes into which the data is mapped
- Hidden layer(s): = in practice, one or two hidden layers are used
- Usually first hidden layer has more nodes
- And second hidden layer has fewer nodes

Input Image

Neural Network Classification



Texture Analysis

Texture Analysis

- Two primary issues in texture analysis:

  - ***texture classification***
  - ***texture segmentation***

- ***Texture classification*** is concerned with identifying a given textured region from a given set of texture classes.

  Each of these regions has unique texture characteristics.

  Statistical methods are extensively used.

- ***Texture segmentation*** is concerned with automatically determining the boundaries between various texture regions in an image.

Texture Classification

- ***Texture classification*** is concerned with identifying a given textured region from a given set of texture classes.
- Each of these regions has unique texture characteristics.
- Statistical methods are extensively used.

Directionality of Texture

Texture is a strong directional feature

e.g., horizontal stripes and vertical stripes are clearly perceived separately

Some texture features can provide directional information

Texture is highly scale dependent – features at one resolution may not exist at a different resolution

Low resolution images generally depend on spectral information, very high resolution images largely depend on spatial details, medium resolution images on texture

Definition of GLCM

- The **GLCM** is defined by:

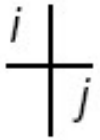
$$P_d(i,j) = n_{i,j} = \#\{f(m,n) = i,\ f(m+dx,\ n+dy) = j;\ 1 \le m \le M;\ 1 \le n \le N\}$$

– where $\mathbf{n_{ij}}$ is the number of occurrences of the pixel values **(i,j)** lying at distance **d** in the image.

– The co-occurrence matrix $\mathbf{P_d}$ has dimension **n× n**, where n is the number of gray levels in the image.

Example

2	1	2	0	1
0	2	1	1	2
0	1	2	2	0
1	2	2	0	1
2	0	1	0	1



$$P_d = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 3 & 2 \end{bmatrix}$$

i: 0, 1, 2

j: 0, 1, 2

There are 16 pairs of pixels in the image which satisfy this spatial separation. Since there are only three gray levels – 0,1,2, **P[i,j]** is a **3×3** matrix.

Algorithm to construct GLCM

Count all pairs of pixels in which the first pixel has a value ***i***, and its matching pair displaced from the first pixel by **d** has a value of ***j***.

This count is entered in the $\mathbf{i^{th}}$ row and $\mathbf{j^{th}}$ column of the matrix $\mathbf{P_d[i,j]}$

Note that $\mathbf{P_d[i,j]}$ is not symmetric in **this** form of counting, since the number of pairs of pixels having gray levels **[i,j]** does not necessarily equal the number of pixel pairs having gray levels **[j,i].**

Normalized GLCM

The elements of $\mathbf{P_d[i,j]}$ can be normalized by dividing each entry by the total number of pixel pairs.

Normalized GLCM **N[i,j],** defined by:

$$N[i,j] = \frac{P[i,j]}{\sum_i \sum_j P[i,j]}$$

which normalizes the co-occurrence values to lie between 0 and 1, and allows them to be thought of as probabilities.

Numeric Features from GLCM

Gray level co-occurrence matrices capture properties of a texture but they are not directly useful for further analysis, such as the comparison of two textures.

Numeric features are computed from the co-occurrence matrix that can be used to represent the texture more compactly.

Haralick Texture Features

Haralick et al. suggested a set of 14 textural features which can be extracted from the co-occurrence matrix, and which contain information about image textural characteristics such as homogeneity, linearity, and contrast.

Haralick, R.M., K. Shanmugam, and I. Dinstein, "Textural features for image classification" IEEE Transactions on Systems, Man and Cybernetics: pp. 610-621. 1973.

Features from GLCM: Angular Second Moment (ASM)

- Angular Second Moment ASM
- ASM = $\sum_{i=1}^{K}\sum_{j=1}^{K} P_d^2(i,j)/R$
- R is a normalizing factor
- ASM is large when only very few gray level pairs are present in the textured image
- K is the number of gray levels

Contrast (CON)

- Contrast CON
- CON = $$\sum_{i=1}^{K}\sum_{j=1}^{K}(i-j)^2 P_d(i,j)/R$$
- This feature highlights co-occurrence of very different gray levels

Entropy (ENT)

- ENT = $$\sum_{i=1}^{K}\sum_{j=1}^{K} P[i,j]\ln\left(\frac{1}{P[i,j]}\right)$$
- ENT emphasises many different co-occurrences
- P(i,j) is the normalized co-occurrence matrix, each entry indicating probability of occurrence of that gray level combination

Inverse Difference Moment (IDM)

- Inverse Difference Moment IDM

- IDM = $$\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{P_d{}^r[i,j]}{|(i-j)^m|_{i\neq j}}$$
- IDM emphasises co-occurrence of close gray levels compared to highly different graylevels. m and r can user specified

Correlation (COR)

- Correlation

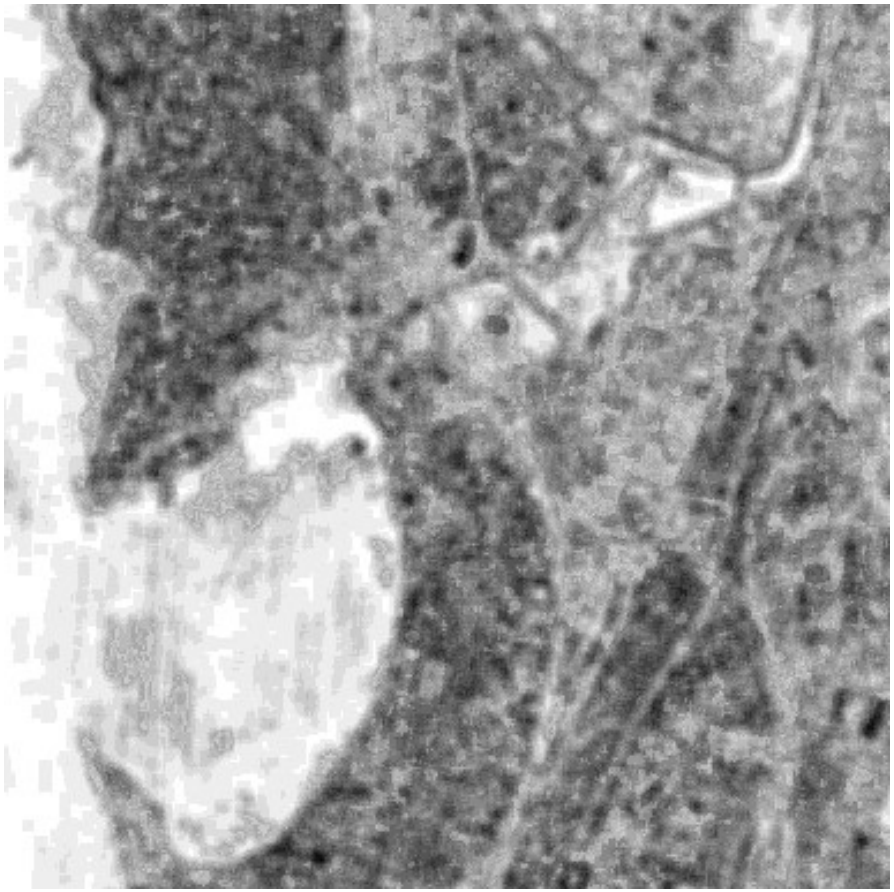
- COR = $$\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{(i-\mu_i)(j-\mu_j)P_d(i,j)/R}{\sigma_i\sigma_j}$$

- COR emphasizes correlation between gray levels i and j.

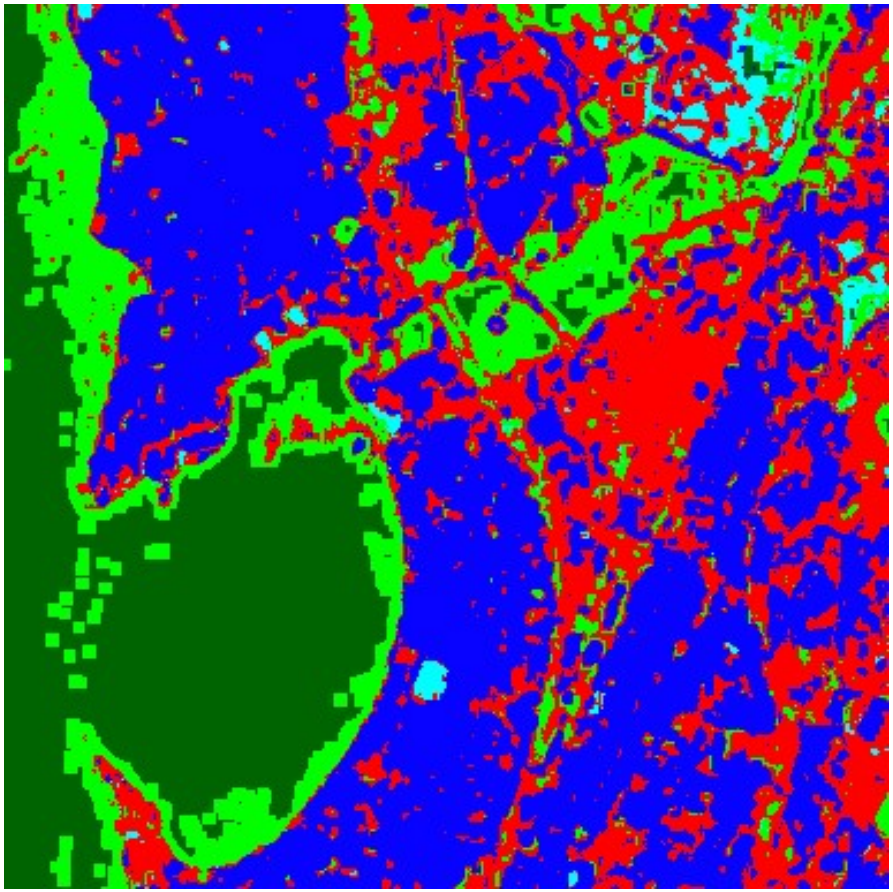
Algorithm for image segmentation

- Specify a window of size wxw
- For the pixels in the window, compute the co-occurrence matrix
- Derive the texture features from the co-occurrence matrix
- Move the window by 1 pixel, and repeat the procedure
- The procedure leads to texture images that may be treated like additional bands, equal to the number of features computed.

Input Image



IDM
Feature



CLASSIFIED IMAGE (Mumbai)

CLASSIFIED IMAGE (Mumbai)

LEGEND

WATER

MARSHY LAND / SHALLOW WATER

HIGHLY BUILT-UP AREA

PARTIALLY BUILT-UP AREA

OPEN AREAS/ GROUNDS

Feature Extraction

Corners and Histogram of Oriented Gradients

Feature Descriptors

1. Harris Corner Detector
2. Histogram of Oriented Gradients (HoG)
3. Scale Invariant Feature Transform (SIFT)

from Rick Szeliski's lecture notes, and other sources…

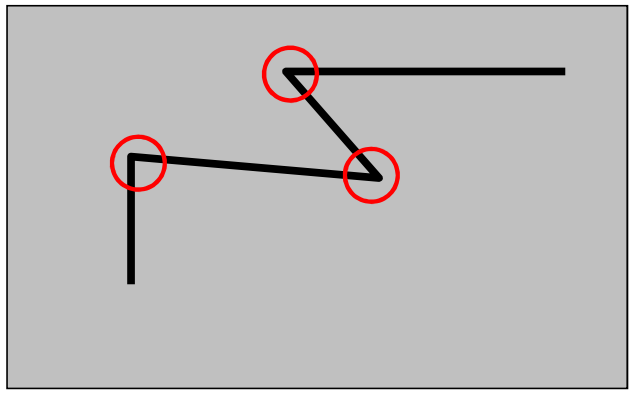
Feature Descriptors

- Images are recognized, matched using some key features that are perceived by humans invariant to rotation, translation, scale, illumination conditions, …
- Some of these abilities are captured in feature descriptors with varying capabilities
- Prominent among them are Harris Corner Detector (Harris-Stevenson algorithm), Histogram of Oriented Gradients (Navneet Dalal and Triggs) and Scale Invariant Feature Transform (David Lowe)

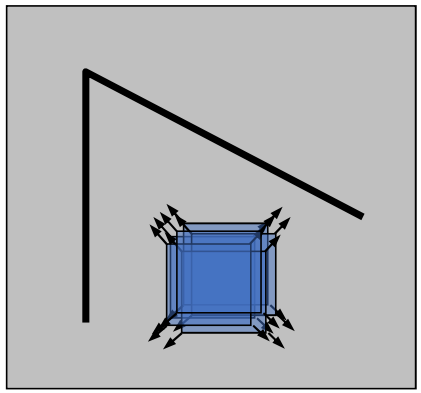
Harris Corner Detector

Harris corner detector

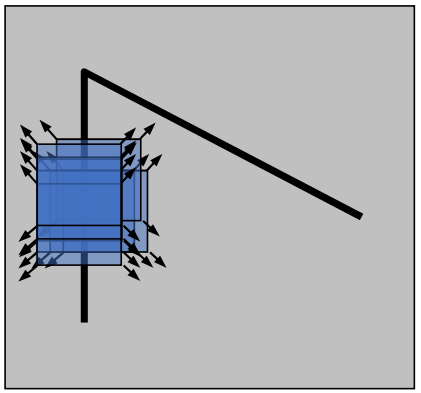
- C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988



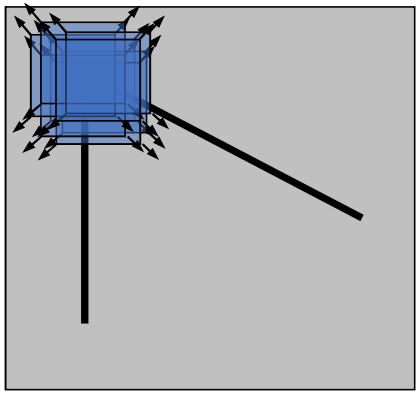
Harris Detector: Basic Idea



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

Harris Detector: Mathematics

Change of intensity for the shift $[u,v]$:

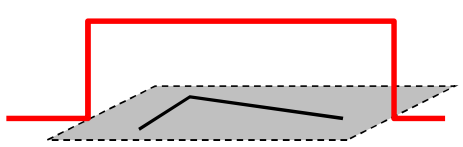
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

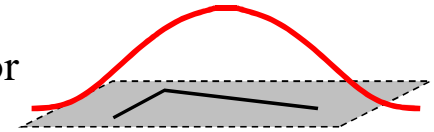
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Taylor series approximation to shifted image

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= \sum_{x,y} w(x,y) \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Can be verified easily by multiplying 1x2 vector with 2x2 matrix with 2x1 vector resulting in a scalar

Harris Detector: Mathematics

For small shifts $[u,v]$ we have a *bilinear* approximation:

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

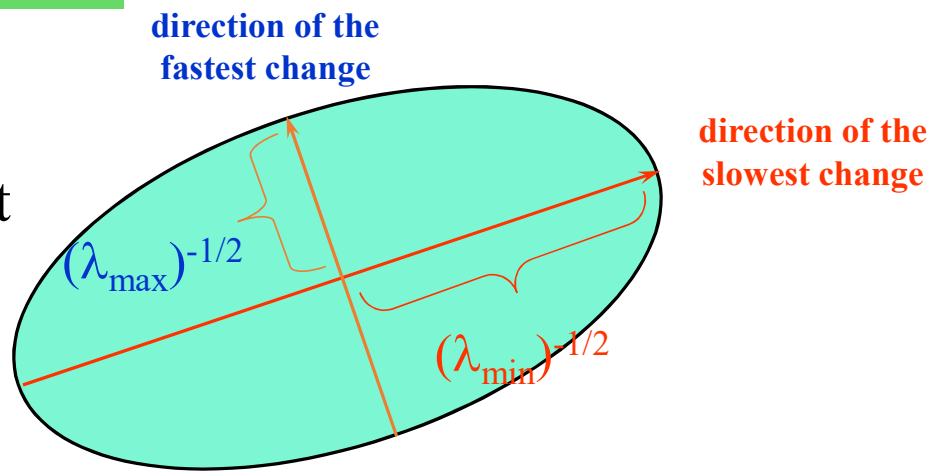
Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



Harris Detector: Threshold

Measure of corner response:

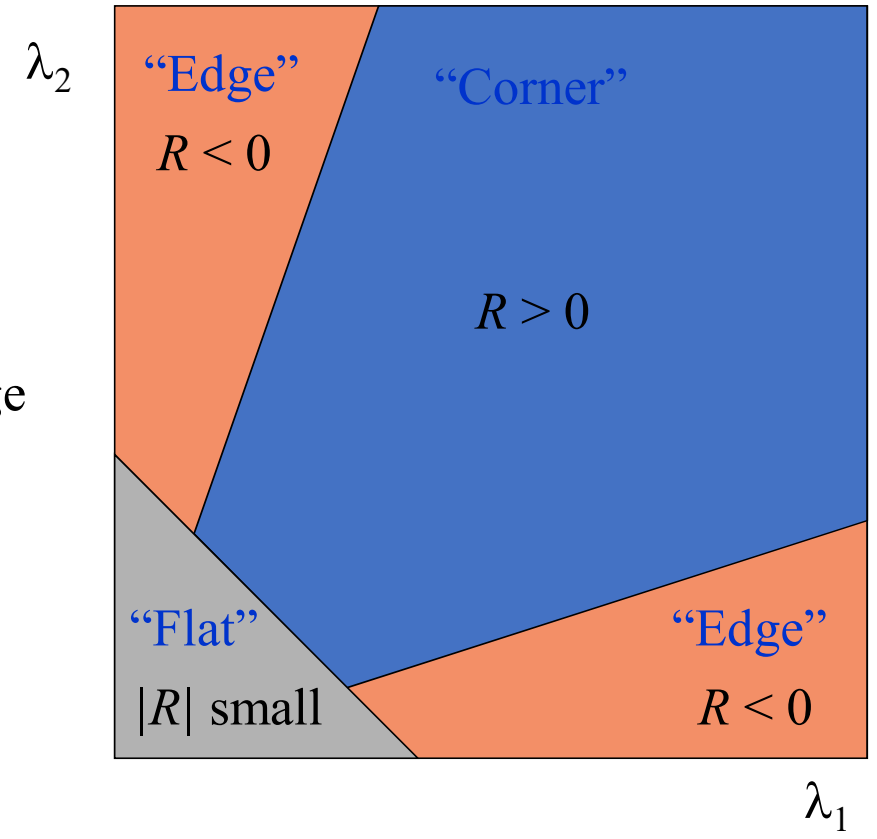
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(k – empirical constant, k = 0.04-0.06)

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



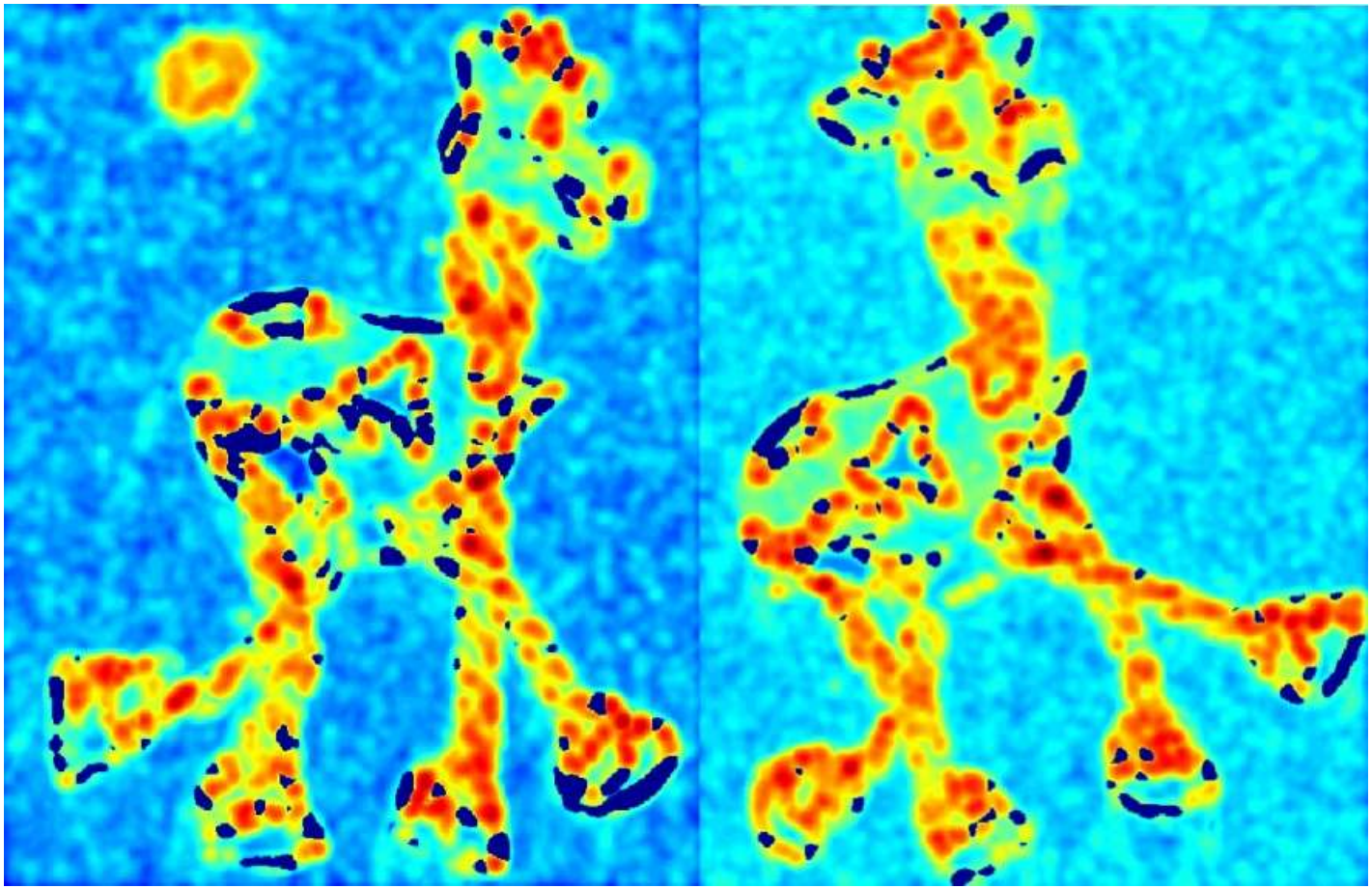
Harris Detector

- The Algorithm:
  - Find points with large corner response function R (R > threshold)
  - Take the points of local maxima of R

Harris Detector: Workflow

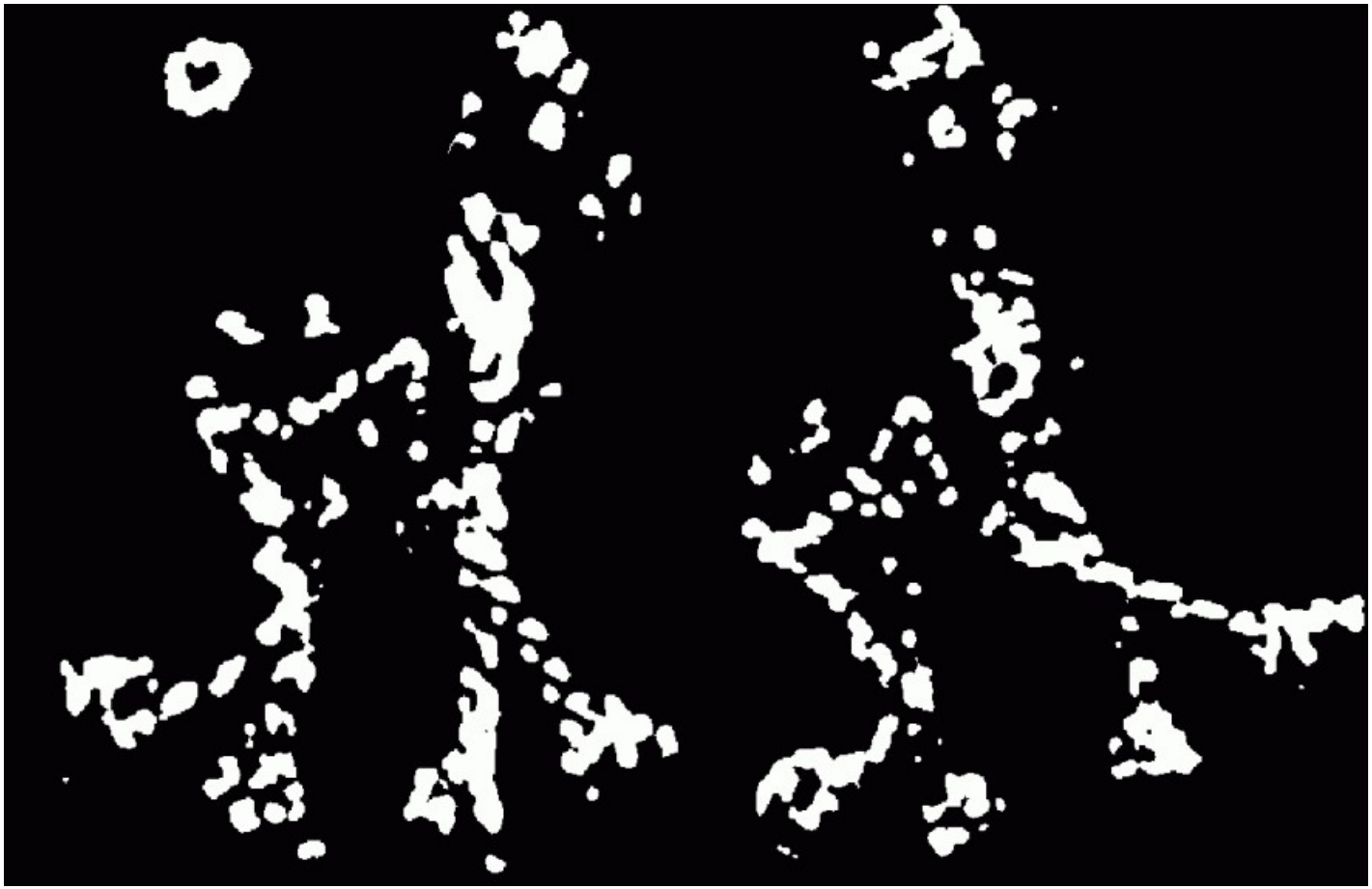
Harris Detector: Workflow

Compute corner response R



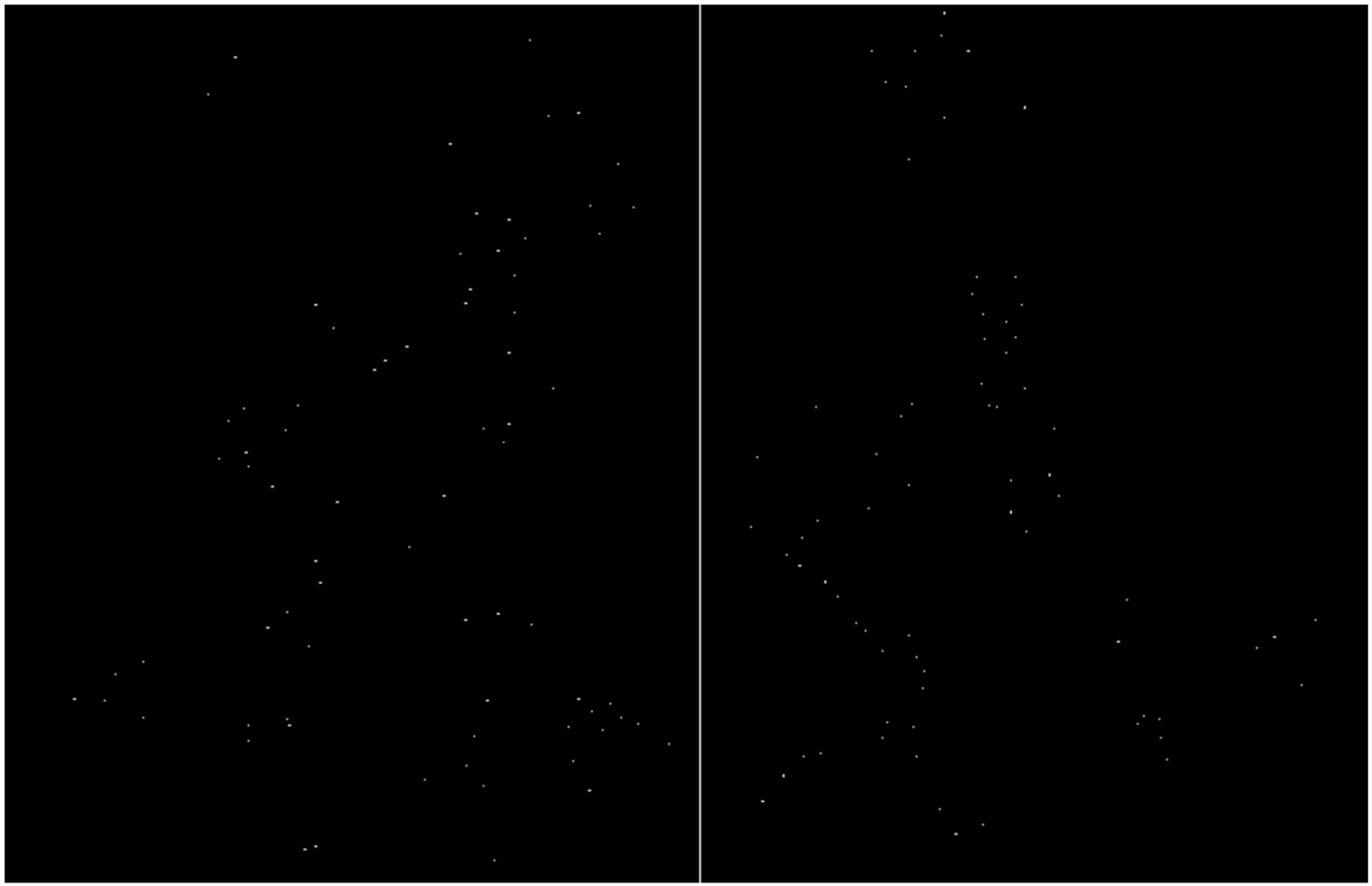
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Summary

- Average intensity change in direction $[u,v]$ can be expressed as a bilinear form:

$$E(u,v) \cong \begin{bmatrix} u,v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

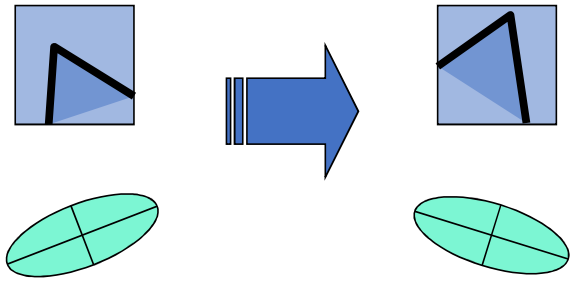
- Describe a point in terms of eigenvalues of M: *measure of corner response*

$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

Harris Detector: Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

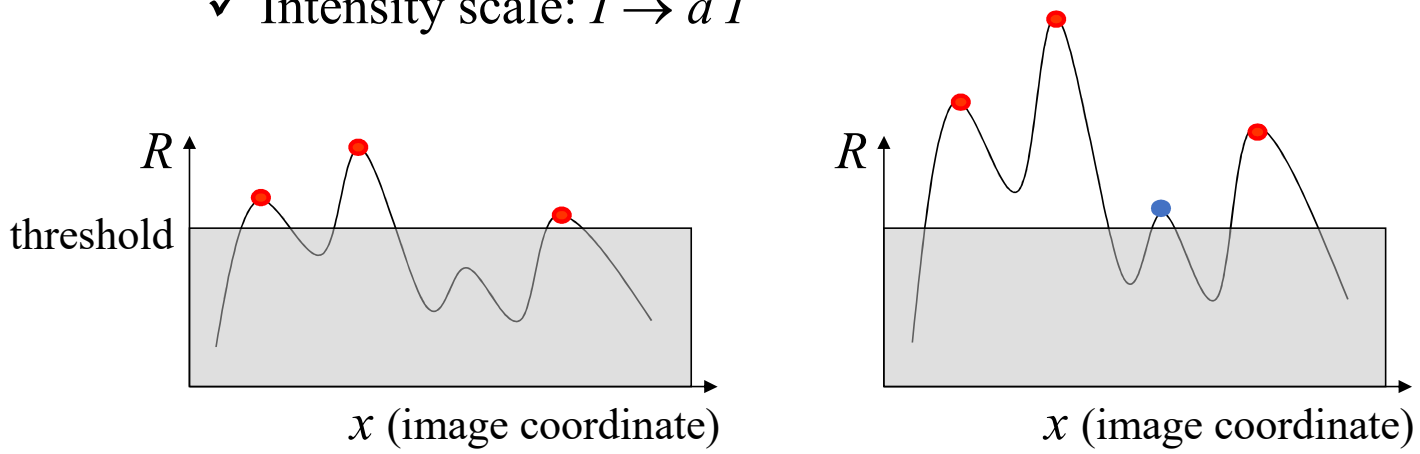
Corner response R is invariant to image rotation

Harris Detector: Some Properties

- Partial invariance to *affine intensity* change

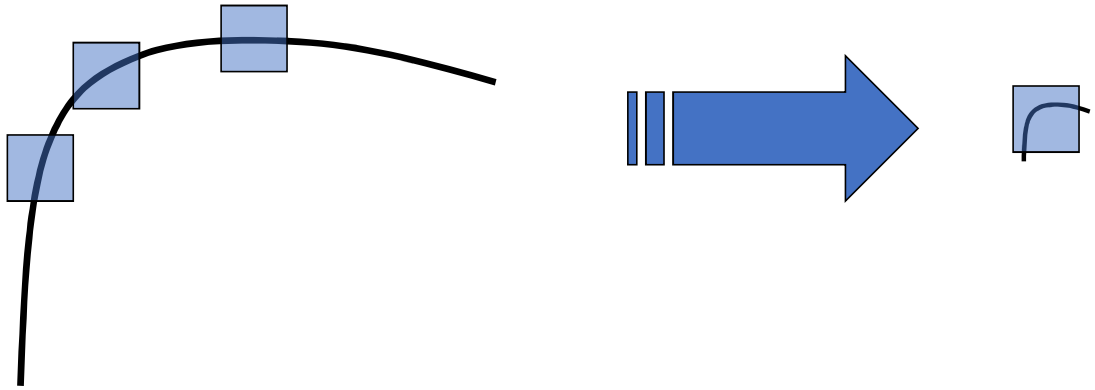
✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

✓ Intensity scale: $I \rightarrow a\, I$



Harris Detector: Some Properties

- But: non-invariant to *image scale*!



All points will be classified as edges

Corner !

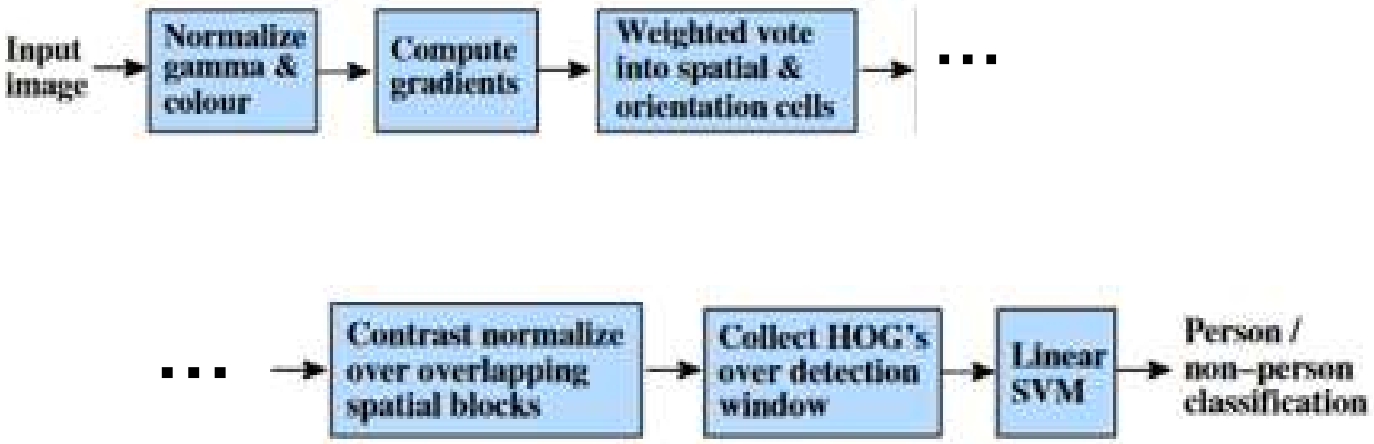
Histogram of Oriented Gradients (HoG)

What is HOG?
(Histograms of Oriented Gradients)

HOG is an edge orientation histogram based descriptor, based on the orientation of the gradient in localized region that is called cells.

Therefore, it is easy to express the rough shape of the object and is robust to variations in geometry and illumination changes.

On the other hand, rotation and scale changes are not supported.



Overall Scheme for HoG based Person-Non-Person Classification

HOG image

HOG feature extraction algorithm

1. The color image is converted to grayscale
2. The luminance gradient is calculated at each pixel
3. To create a histogram of gradient orientations for each cell.
   - Feature quantity becomes robust to changes of form
4. Normalization and Descriptor Blocks
   - Feature quantity becomes robust to changes in illumination

HOG feature extraction algorithm(1)

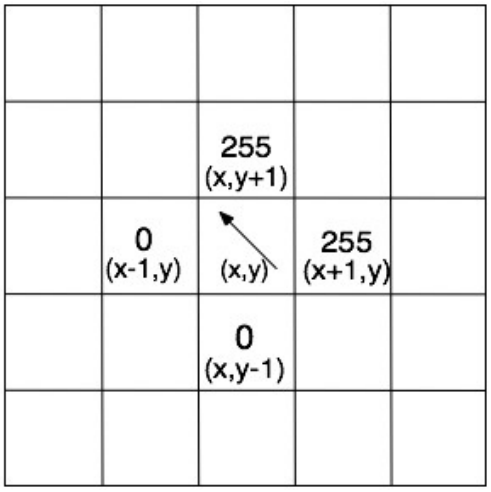
2. The luminance gradient is calculated at each pixel
   - The luminance gradient is a vector with magnitude m and orientation θ represented by the change in the luminance.

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}$$

$$\theta(x,y)=\tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right)$$

$$-\frac{\Pi}{2}<\theta<\frac{\Pi}{2}$$

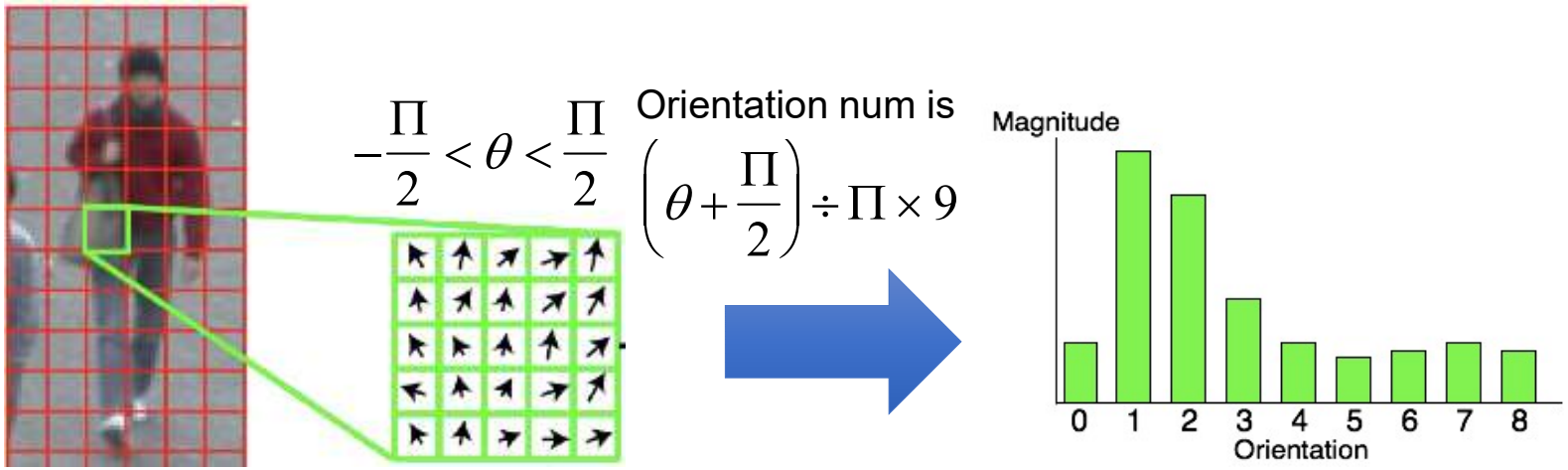
※L is the luminance value of pixel



HOG feature extraction algorithm(2)

3. To create a histogram of gradient orientations for each cell(5 × 5pixel) using the gradient magnitude and orientation of the calculated.
   - The orientation bins are evenly spaced over 0° – 180° with nine bins each of 20° each. By adding the magnitude of the luminance gradient for each orientation, generate a histogram.

Image size = 60x30

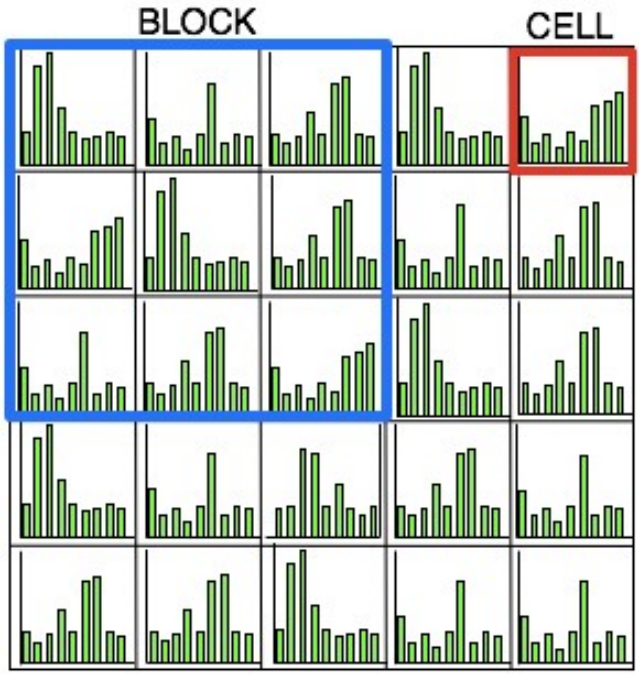


HOG feature extraction algorithm(3)

4. **Normalization and Descriptor Blocks**
   - Normalization is performed using the following equation:

$$v(n) = \frac{v(n)}{\sqrt{\left(\sum_{k=1}^{3\times3\times9} v(k)^2\right) + 1}}$$

v(n) is the magnitude of each direction



HOG feature extraction algorithm(3)

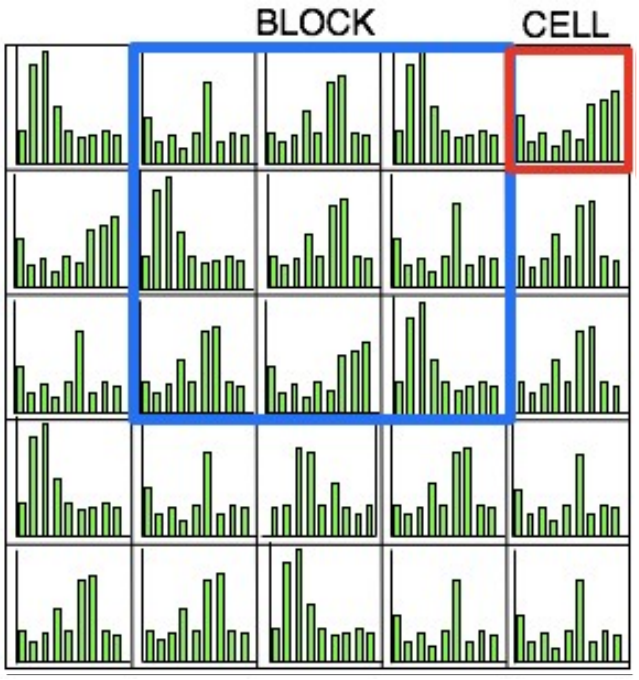
4. **Normalization and Descriptor Blocks**
   - Normalization is performed using the following equation:

$$v(n) = \frac{v(n)}{\sqrt{\left(\sum_{k=1}^{3\times3\times9} v(k)^2\right) + 1}}$$

v is the magnitude of each direction

Normalized vector is computed over Cells and 9 orientations

For a block, the vector size = 9x9=81



HOG image

Feature descriptor size

- 12x6 cells
- Number of orientations = 9
- Block size = 3x3=9
- Block moves 4 steps to right and 10 steps down

Descriptor size for total image = 10x4x9x9= 3240

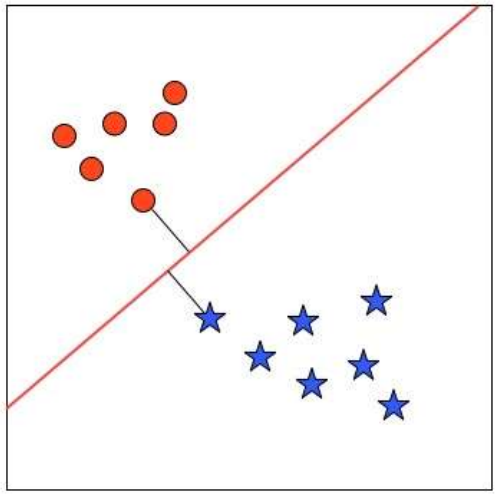
Example of using HOG

- HOG can represent a rough shape of the object, so that it has been used for general object recognition, such as people or cars.
- In order to achieve the general object recognition, the classifier (eg SVM) is be used.
  1. To teach the classifier, the correct image and the incorrect image.
  2. Scan the classifier to determine whether there are people in the detection window.

SVM based Classification

SVM divides space into two domains according to a teacher signal.

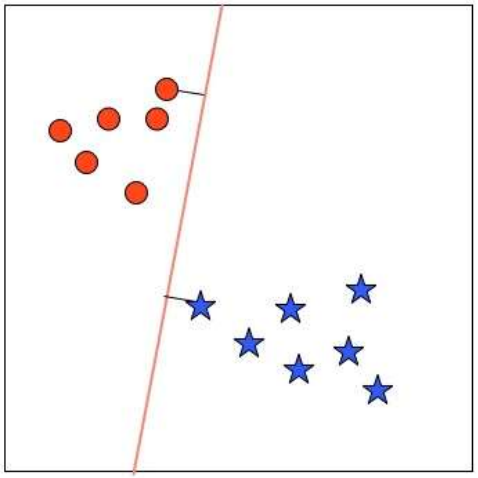
New examples are predicted to belong to a category based on which side of the gap domain.



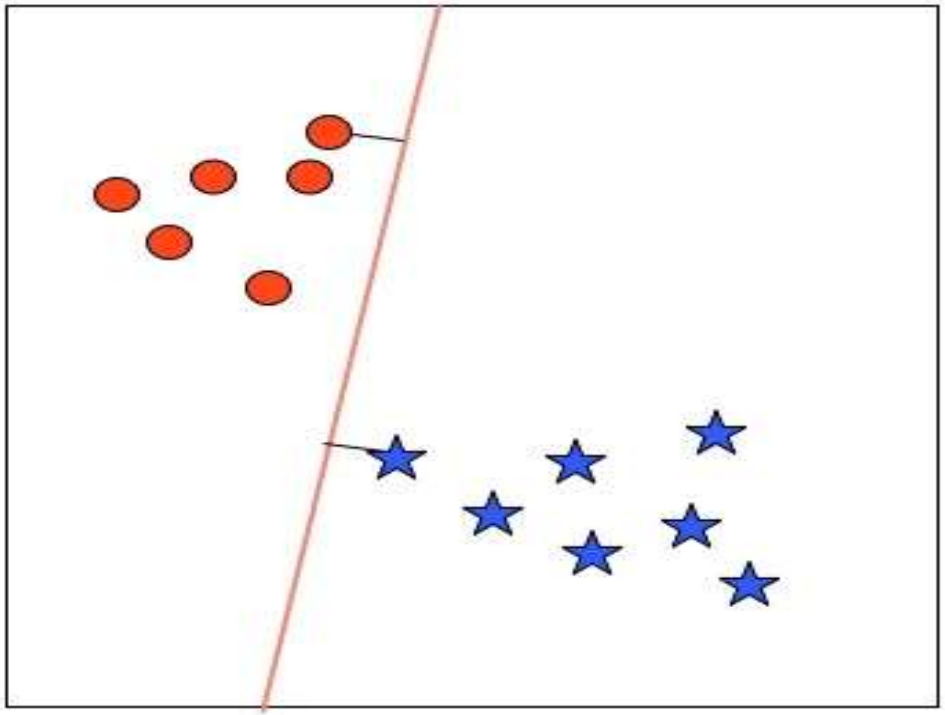
SVM Based Classification

SVM divides space into two domains according to a teacher signal.

New examples are predicted to belong to a category based on which side of the gap domain.



SVM Based Classification

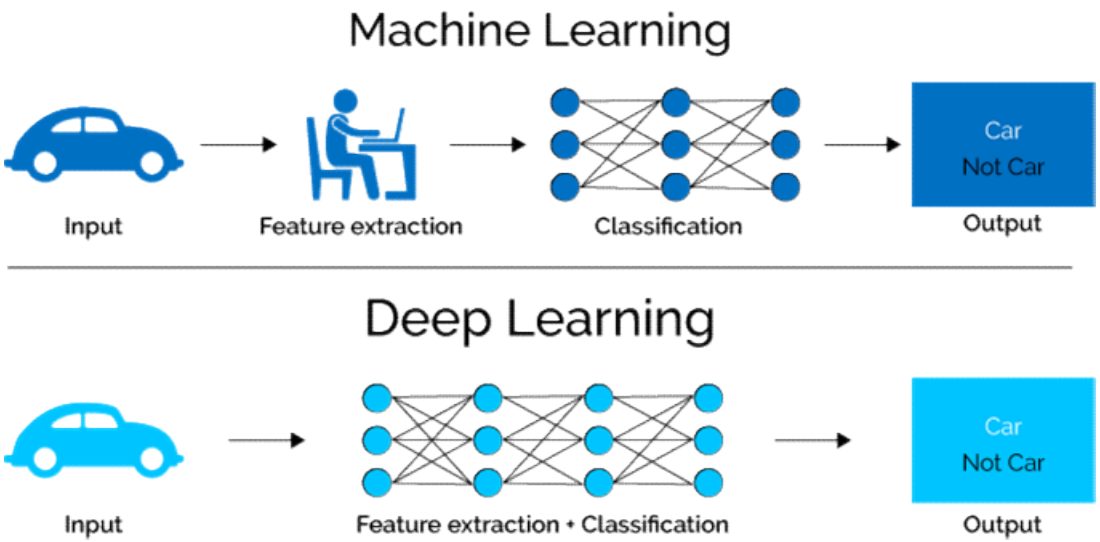


Deep Learning

What is Deep Learning (DL) ?

A machine learning subfield for learning **representations** of data. Exceptionally effective at **learning patterns (spectral and spatial)**.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

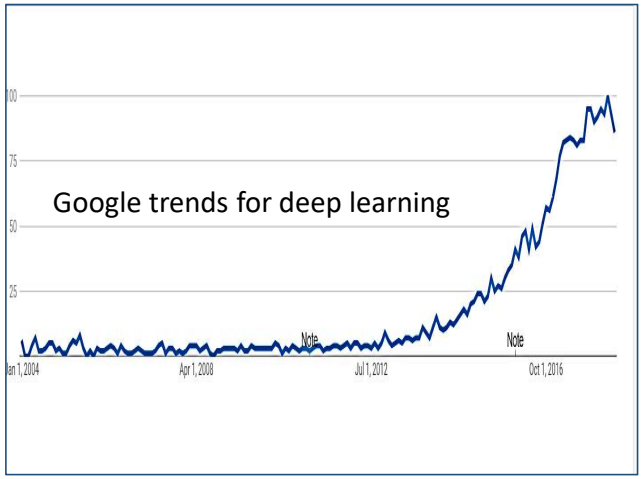


Source: https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Why is DL useful?

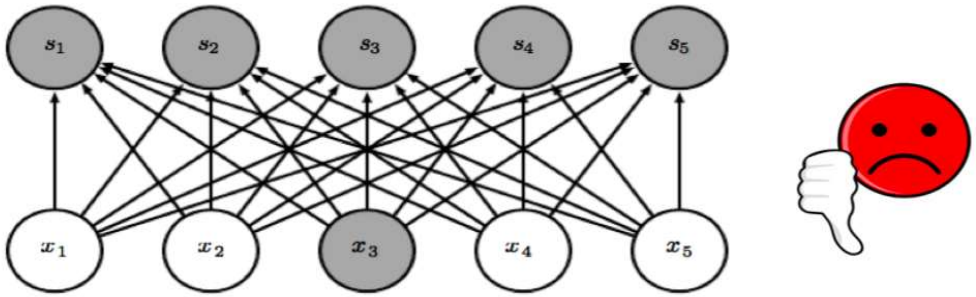
- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate
- Learned Features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible**, (almost?) **universal**, learnable framework for representing world, visual and linguistic information.
- Can learn in both unsupervised and supervised manner
- Effective **end-to-end** joint system learning

In 2010 DL started outperforming other ML techniques
first in speech and vision, then NLP

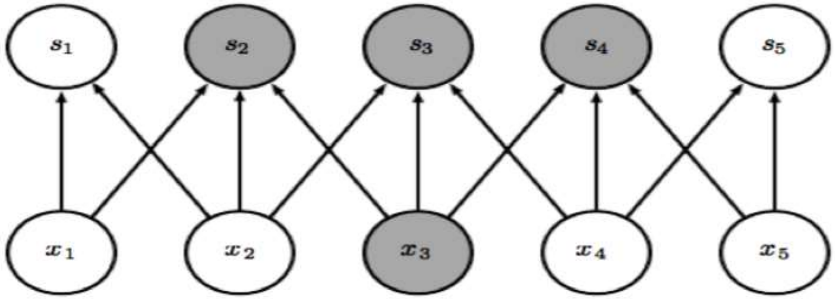


Convolutional Neural Networks

Unlike fully connected networks, CNNs use shared weights to reduce the number of parameters



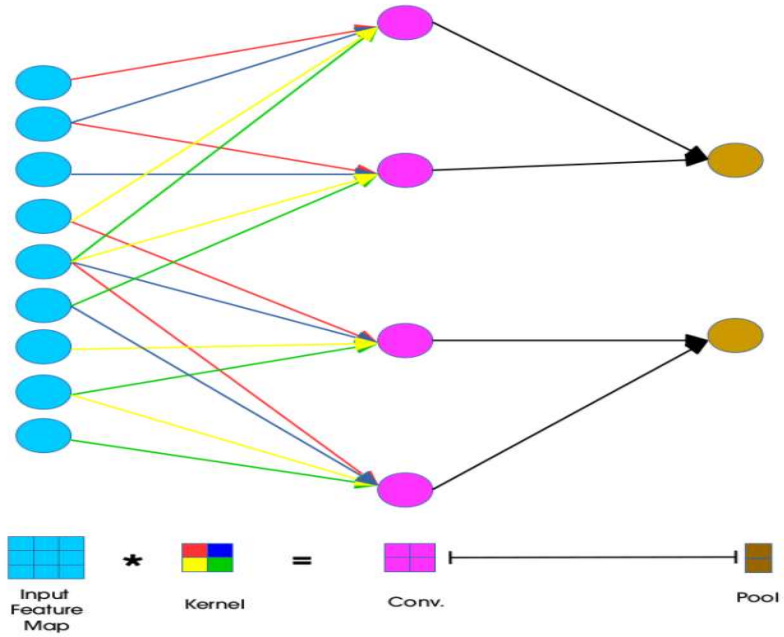
Fully connected Network



Convolutional network

Convolutional Neural Networks

Learned weights simulate convolutional filters



The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

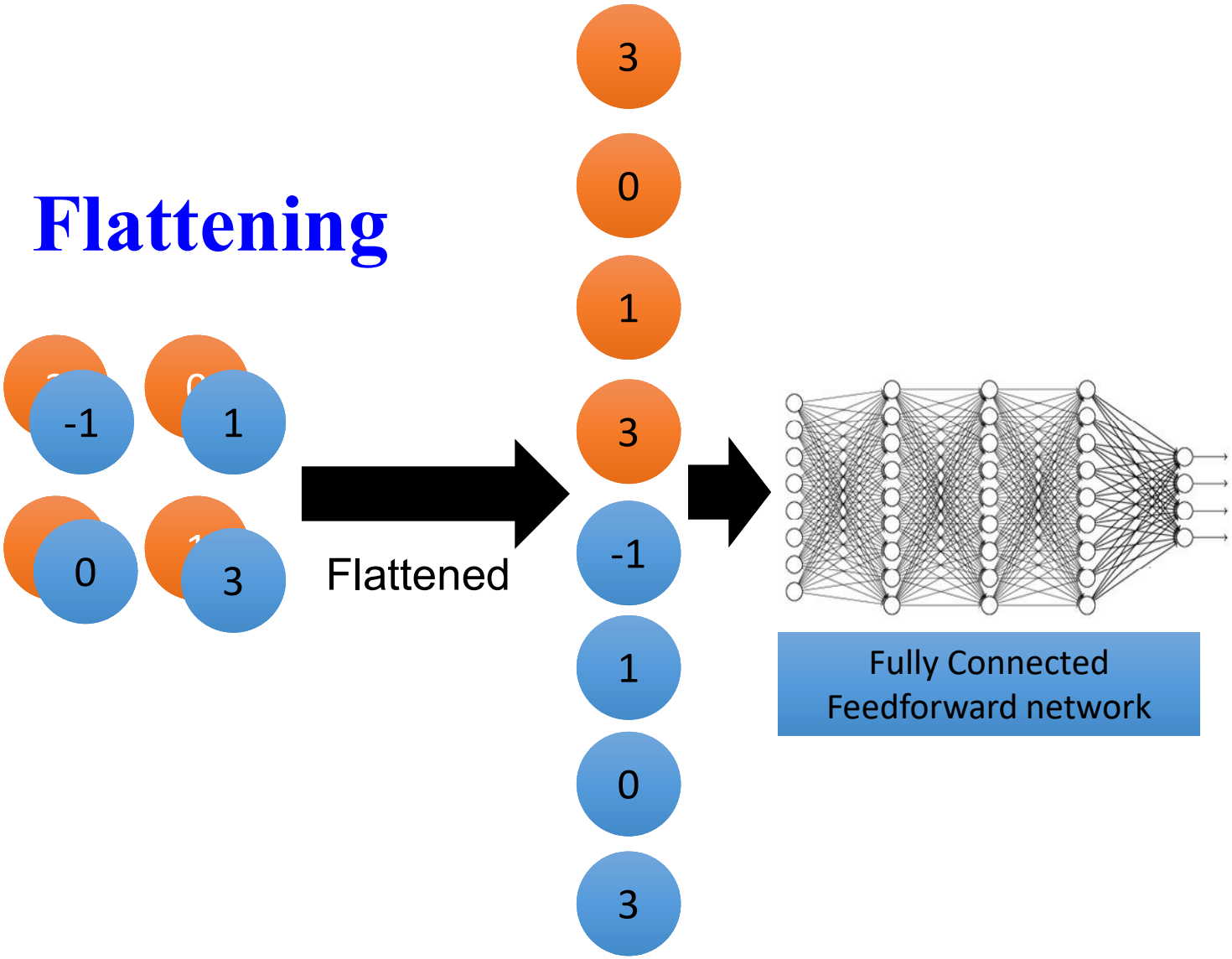
A new image

cat dog ……

Fully Connected Feedforward network

Flattened

Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Popular variants of CNNs

1D CNN: Input and filters are 1D vectors

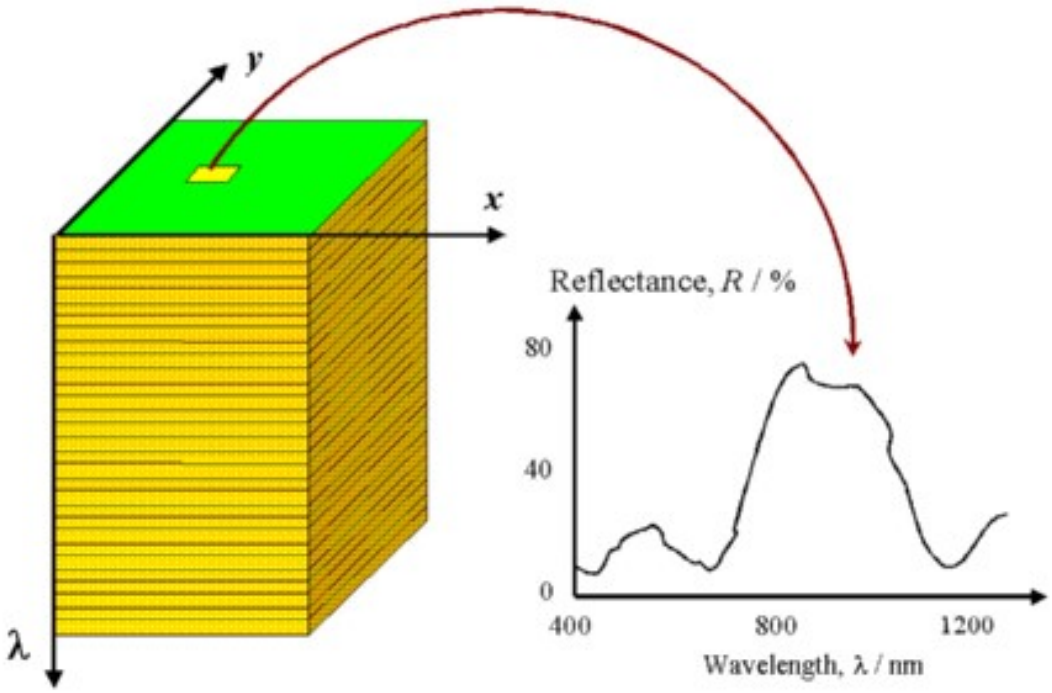
2D CNN: Input and filters are 2D vectors and filters move in the 2D spatial space

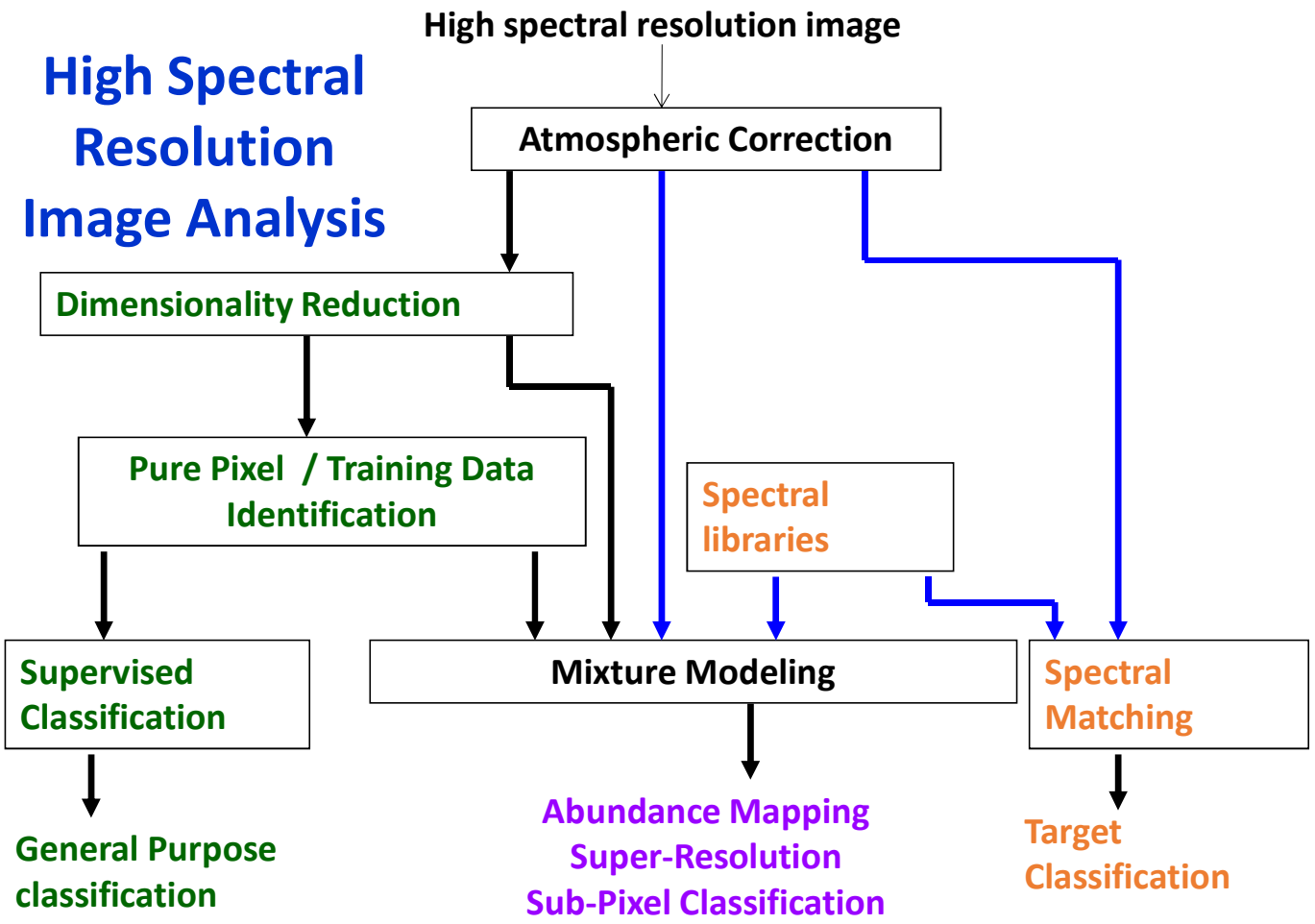
3D CNN: Input and filters are 3D vectors and filters move in spectral and spatial dimension; Feature maps are 3d tensors

Conv. Autoencoders: CNNs in an encoder-decoder framework

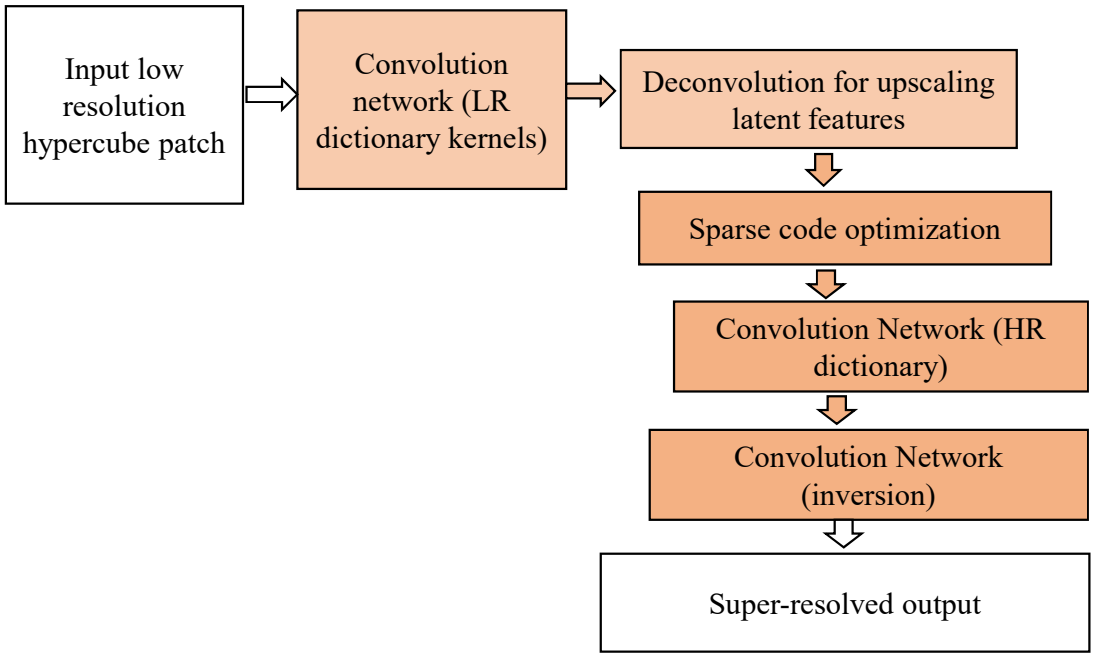
Deep Learning application with High-Dimensional Remotely Sensed Images

Hyperspectral Remote Sensing




High Spectral Resolution Image Analysis
High spectral resolution image
Atmospheric Correction
Dimensionality Reduction
Pure Pixel / Training Data Identification
Spectral libraries
Supervised Classification
Mixture Modeling
Spectral Matching
General Purpose classification
Abundance Mapping
Super-Resolution
Sub-Pixel Classification
Target Classification

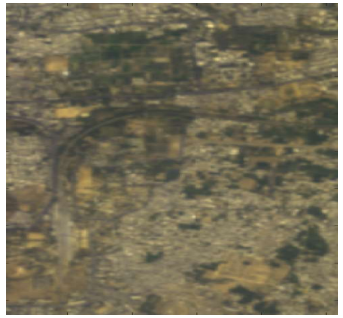
Sample super-resolution framework



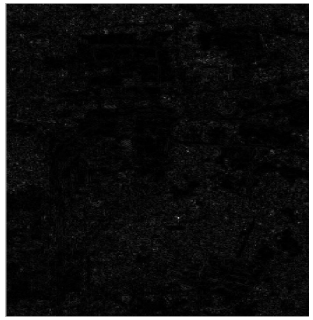
CNN based SR Approach on AVIRIS NG Dataset

a) Original HR image

b) Simulated coarser version

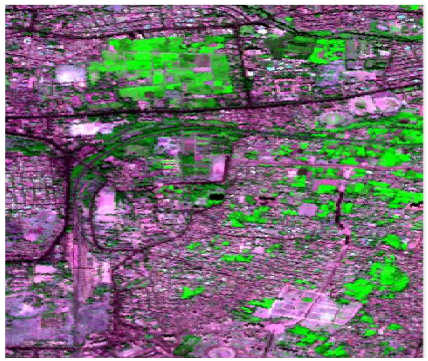


Result

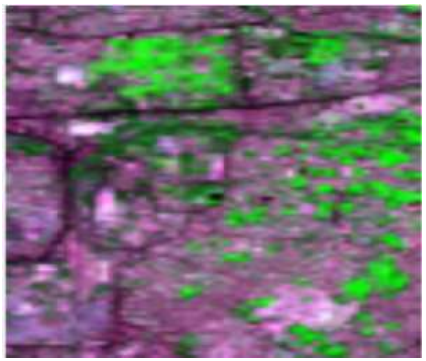


d) Norm of reconstruction error

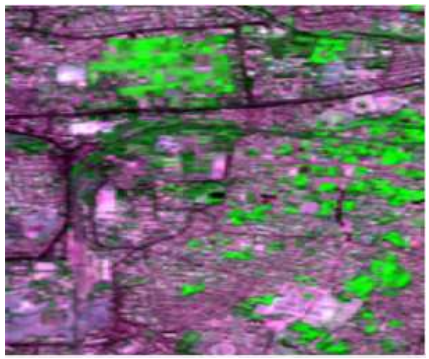
SR approach on AVIRIS Dataset



a) Original HR image

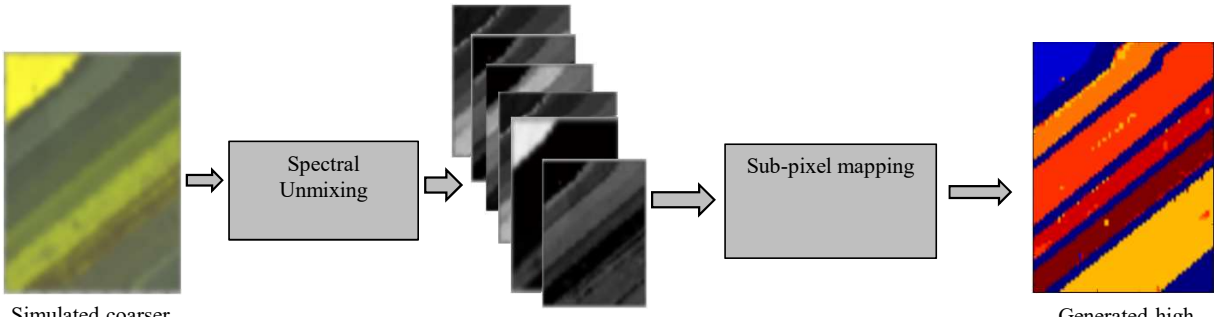


b) Simulated coarser version



Result

Sub-Pixel Mapping



Simulated coarser hypercube (Y)

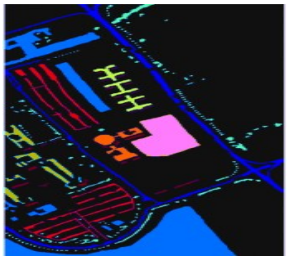
Probability estimates/ Fractional abundance images (A_k) for each class k

Generated high resolution classified image (F)

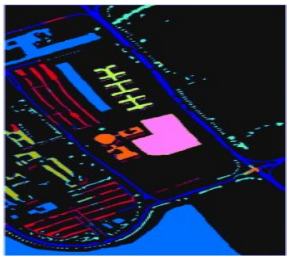
Sub-pixel Mapping Example-1

a) Original HR image

b) Simulated coarser version (Z=3)



c) HR ground truth

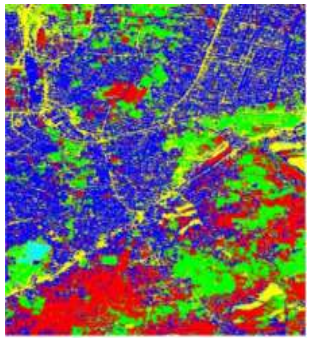


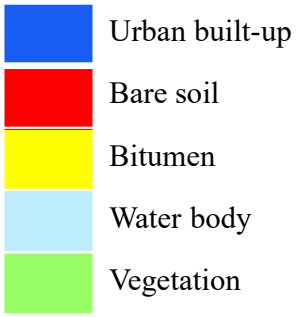
c) Sub-pixel level classified image (architecture-2)

Sub-pixel Mapping Example-2

a) Original HR image

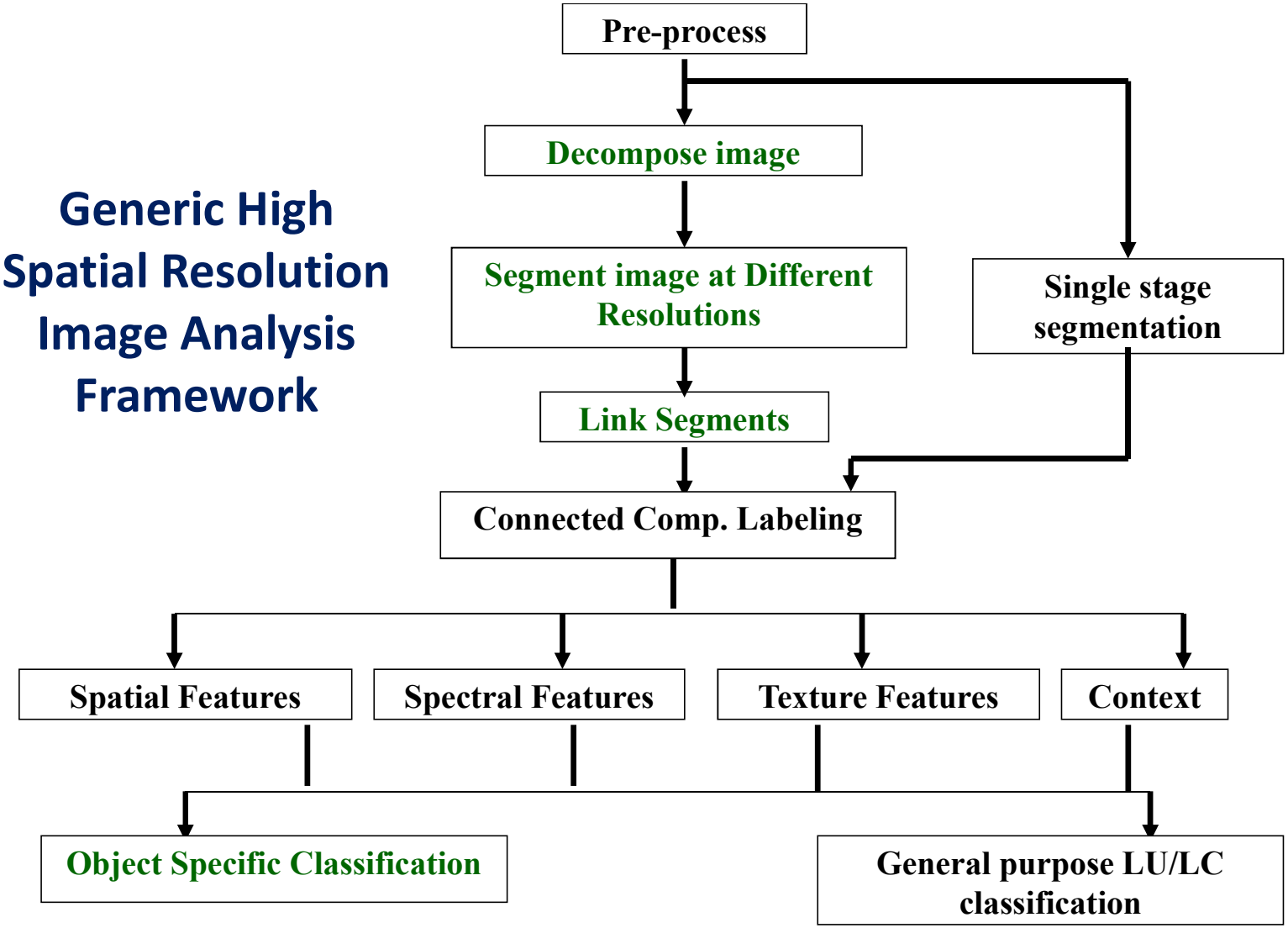
b) Simulated coarser version (Z=4)





c) Sub-pixel level classified image (architecture-2)

Generic High Spatial Resolution Image Analysis Framework

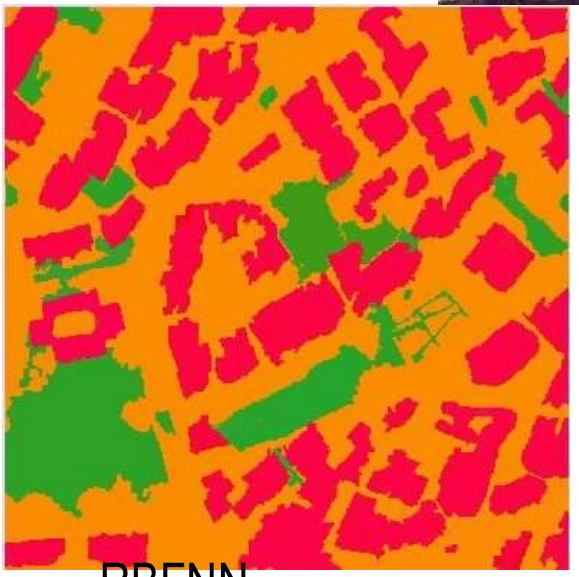


Study Area 2

OB Segmentation

Settlements

Buildup
Open ground
Vegetation

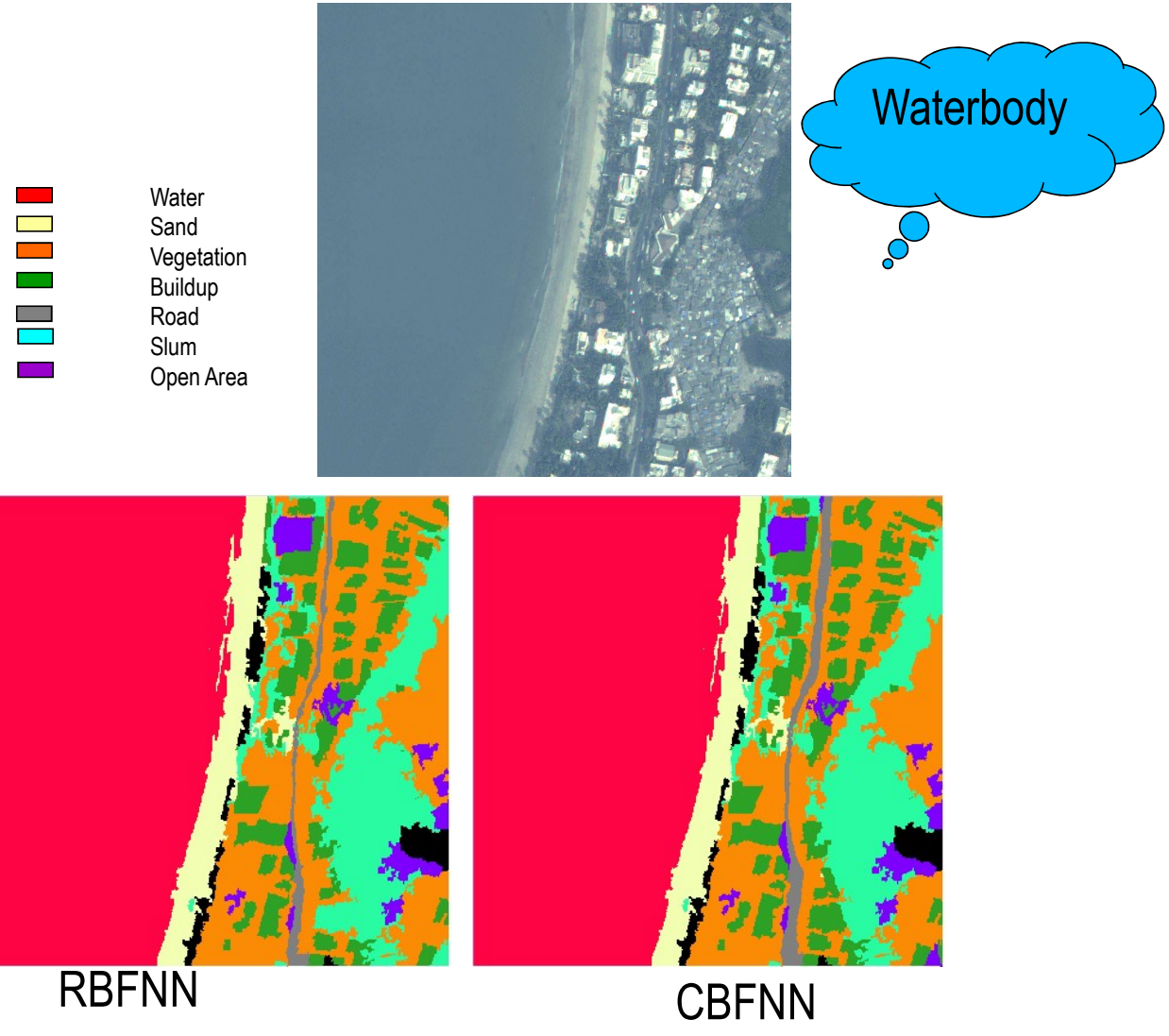


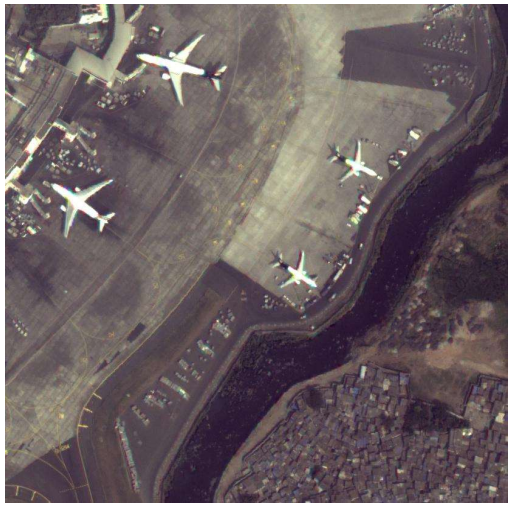
RBFNN

CBFNN

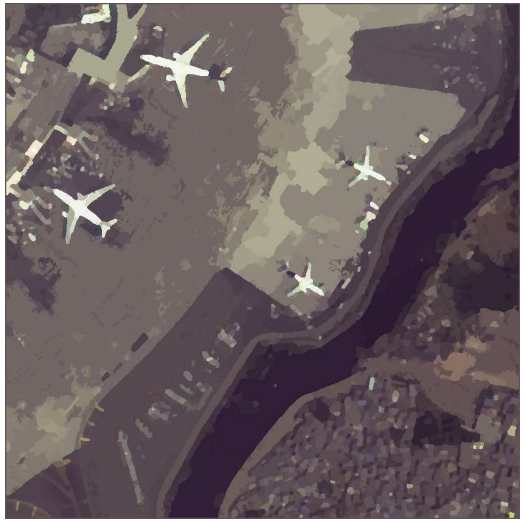
Study Area 4

OB Segmentation


Water
Sand
Vegetation
Buildup
Road
Slum
Open Area
Waterbody
RBFNN
CBFNN



Study Area 6



OB Segmentation


Aero plane
Open Area
Vegetation
Water
Shadow
Settlements
Vehicle
Road
Open Area
RBFNN
CBFNN

Target detection

Finally…

The success of machine learning, in any domain of application, depends on three key factors:

- Reliable and adequate training data
- Features that adequately represent the classes
- Learning algorithm that can fully capture the characteristics of the classes from the features and training data so that it can generalize well

If anyone of these three is weak, the machine learning method results in a sub-optimal solution to the problem at hand

Questions?