Machine Learning

Machine Learning

according to Google

- The ability of a machine to improve its performance based on previous results.

- The process by which computer systems can be directed to improve their performance over time.

- Subspecialty of artificial intelligence concerned with developing methods for software to learn from experience or extract knowledge from examples in a database.

- The ability of a program to learn from experience — that is, to modify its execution on the basis of newly acquired information.

- Machine learning is an area of artificial intelligence concerned with the development of techniques which allow computers to "learn". More specifically, machine learning is a method for creating computer programs by the analysis of data sets. Machine learning overlaps heavily with statistics, since both fields study the analysis of data, but unlike statistics, machine learning is concerned with the algorithmic complexity of computational implementations. ...

Some Examples

- ZIP code recognition
- Loan application classification
- Signature recognition
- Voice recognition over phone
- Credit card fraud detection
- Spam filter
- Collaborative Filtering: suggesting other products at Amazone.com
- Marketing
- Stock market prediction
- Expert level chess and checkers systems
- biometric identification (fingerprints, DNA, iris scan, face)
- machine translation
- web-search
- document & information retrieval
- camera surveillance
- robosoccer
- and so on and so on...

Why is this cool/important?

- Modern technologies generate data at an unprecedented scale.
  - The amount of data doubles every year.

“One petabyte is equivalent to the text in one billion books,
yet many scientific instruments, including the Large Synoptic Survey Telescope,
will soon be generating several petabytes annually”.

(2020 Computing: Science in an exponential world: *Nature* Published online: 22 March 2006)

- Computers dominate our daily lives
  - Science, industry, army, our social interactions etc.

We can no longer “eyeball” the images captured by some satellite for interesting events, or check every webpage for some topic.

We need to trust computers to do the work for us.

Types of Learning

We will be concerned with these topics in thi sclass

- Supervised Learning
  - Labels are provided, there is a strong learning signal.
  - e.g. classification, regression.

- Semi-supervised Learning.
  - Only part of the data have labels.
  - e.g. a child growing up.

- Reinforcement learning.
  - The learning signal is a (scalar) reward and may come with a delay.
  - e.g. trying to learn to play chess, a mouse in a maze.

- Unsupervised learning
  - There is no direct learning signal. We are simply trying to find structure in data.
  - e.g. clustering, dimensionality reduction.

Ingredients

- Data:
  - what kind of data do we have?
- Prior assumptions:
  - what do we know a priori about the problem?
- Representation:
  - How do we represent the data?
- Model / Hypothesis space:
  - What hypotheses are we willing to entertain to explain the data?
- Feedback / learning signal:
  - what kind of learning signal do we have (delayed, labels)?
- Learning algorithm:
  - How do we update the model (or set of hypothesis) from feedback?
- Evaluation:
  - How well did we do, should we change the model?

Supervised Learning I

Example: Imagine you want to classify versus

Data: 100 monkey images and 200 human images with labels what is what.

$$\{\vec{x}_{i,} y_i = 0\},\ \ i = 1,\ldots,100$$

$$\{\vec{x}_{j,} y_j = 1\},\ \ j = 1,\ldots,200$$

where x represents the greyscale of the image pixels and
y=0 means “monkey” while y=1 means “human”.

Task: Here is a new image: monkey or human?

1 nearest neighbors

(your first ML algorithm!)

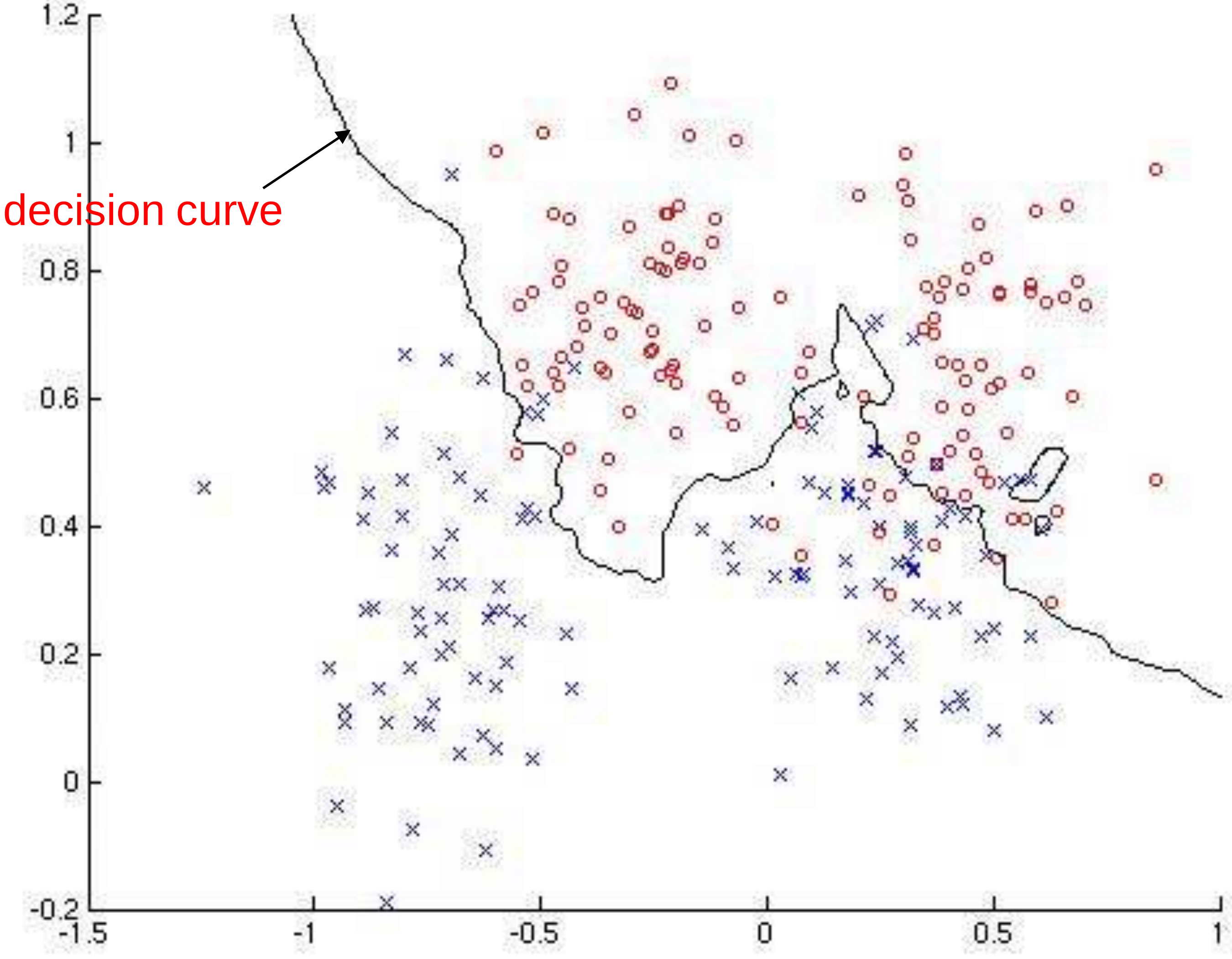
Idea:

1. Find the picture in the database which is closest your query image.
2. Check its label.
3. Declare the class of your query image to be the same as that of the closest picture.

query

closest image

1NN Decision Surface



Distance Metric

- How do we measure what it means to be “close”?
- Depending on the problem we should choose an appropriate distance metric.

Hamming distance:

$$D(\vec{x}_n, \vec{x}_m) = | \vec{x}_n - \vec{x}_m | \qquad \{x = \text{discrete}\};$$

Scaled Euclidean Distance:

$$D(\vec{x}_n, \vec{x}_m) = (\vec{x}_n - \vec{x}_m)^T A (\vec{x}_n - \vec{x}_m) \qquad \{x = cont.\};$$

Remarks on NN methods

- We only need to construct a classifier that works locally for each query. Hence: We don’t need to construct a classifier everywhere in space.

- Classifying is done at query time. This can be computationally taxing at a time where you might want to be fast.

- Memory inefficient (you have to keep all data around).

- Curse of dimensionality: imagine many features are irrelevant / noisy → distances are always large.

- Very flexible, not many prior assumptions.

- k-NN variants robust against “bad examples”.

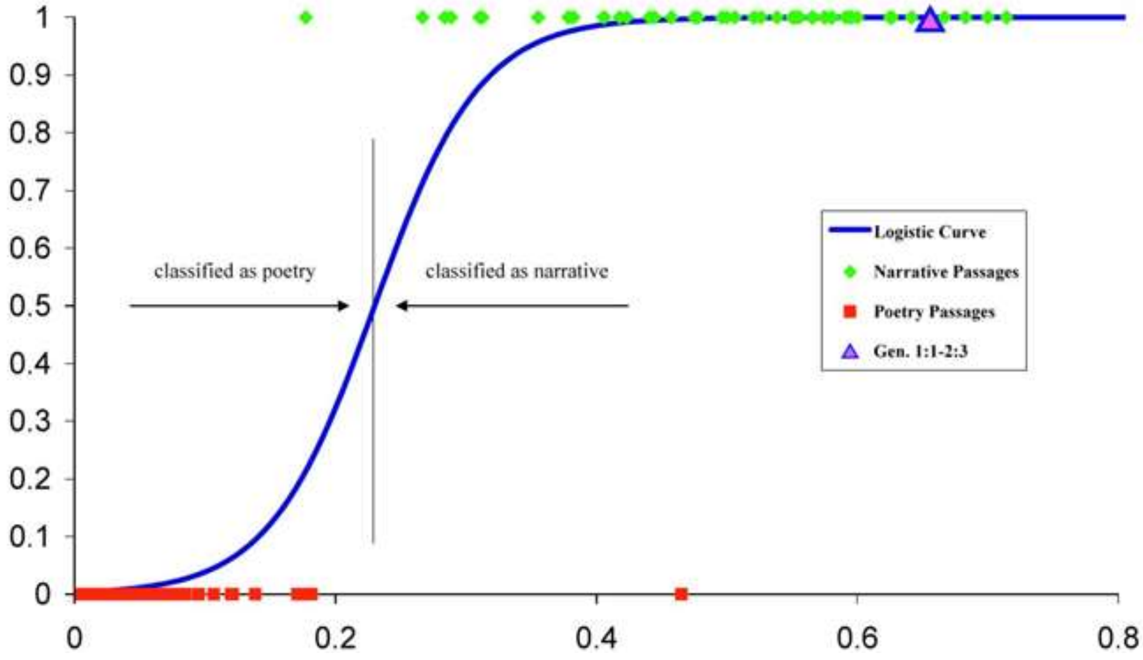
Non-parametric Methods

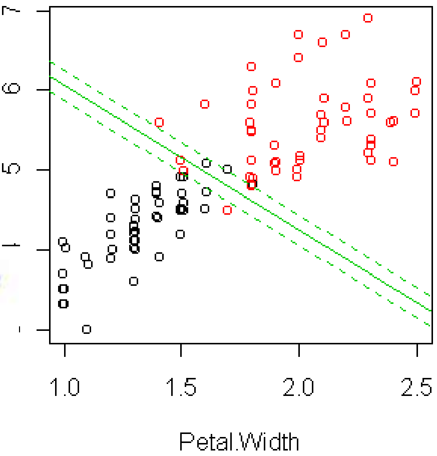
- Non-parametric methods keep all the data cases/examples in memory.
- A better name is: “instance-based” learning
- As the data-set grows, the complexity of the decision surface grows.
- Sometimes, non-parametric methods have some parameters to tune...
- Very few assumptions (we let the data speak).

Logistic Regression / Perceptron

(your second ML algorithm!)

- Fits a soft decision boundary between the cla

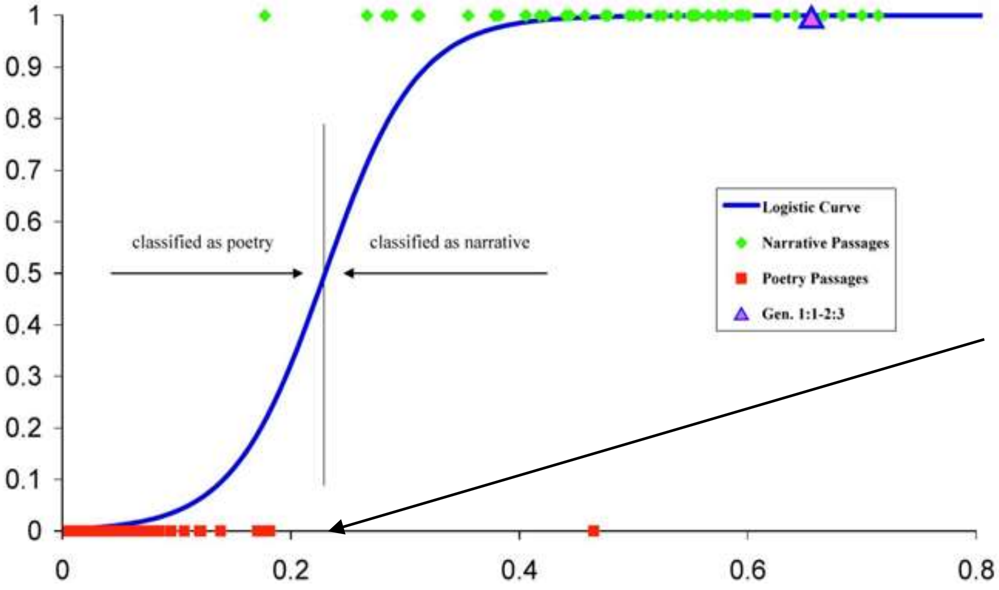




1 dimension

2 dimensions

The logit / sigmoid



$$h(X) = \frac{1}{1 + \exp[-(W^T X + b)]}$$

Determines the offset

Determines the angle
and the steepness.

Objective

- We interpret h(x) as the probability of classifying a data case as positive.
- We want to maximize the total probability of the data-vectors:

$$O = \sum_{\substack{\text{positive}\\ \text{examples}\\ (y_n=1)}} \log\left[h(x_n)\right] + \sum_{\substack{\text{negative}\\ \text{examples}\\ (y_n=0)}} \log\left[1 - h(x_n)\right]$$

Algorithm in detail

- Repeat until convergence (gradient descend):

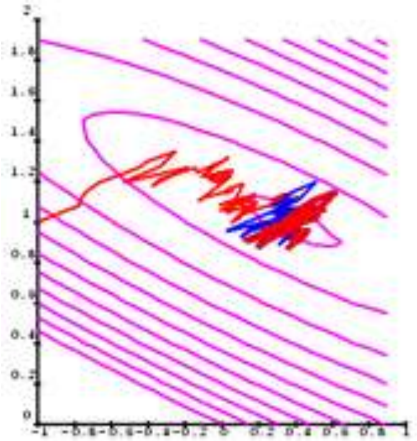
$$W \leftarrow W + \eta \frac{\partial O}{\partial W}$$

$$\frac{\partial O}{\partial W} = \sum_{\substack{positive\\examples\\(y_n=1)}} \left(1 - f(x_n)\right) x_n - \sum_{\substack{negative\\examples\\(y_n=0)}} f(x_n)\ x_n$$

$$b \leftarrow b + \eta \frac{\partial O}{\partial b}$$

$$\frac{\partial O}{\partial b} = \sum_{\substack{positive\\examples\\(y_n=1)}} \left(1 - f(x_n)\right) - \sum_{\substack{negative\\examples\\(y_n=0)}} f(x_n)$$

A Note on Stochastic GD



- For very large problems it is more efficient to compute the gradient using a small (random) subset of the data.

- For every new update you pick a new random subset.

- Towards convergence, you decrease the stepsize.

- Why is this more efficient?

→The gradient is an average over many data-points.

→ If your parameters are very “bad”, every data-point will tell you to move in the same direction, so you need only a few data-points to find that direction.

→Towards convergence you need all the data-points.

→ A small step-size effectively averages over many data-points.

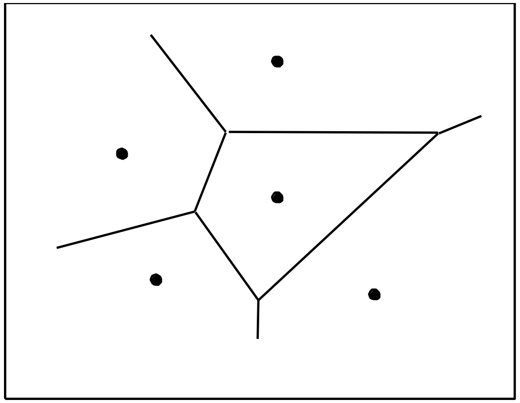
Parametric Methods

- Parametric methods fit a finite set of parameters to the data.
- Unlike NP methods, this implies a maximum complexity to the algorithm.
- "Assumption heavy": by choosing the parameterized model you impose your prior assumptions (this can be an advantage when you have sound assumptions!)
- Classifier is build off-line. Classification is fast at query time.
- Easy on memory: samples are summarized through model parameters.

Hypothesis Space

- An hypothesis h: X→[0,1] for a binary classifier is a function that maps all possible input values to either class 0 or class 1.

- E.g. for 1-NN the hypothesis h(X) is given by: →



- The hypothesis space H, is the space of all hypotheses that you are willing to consider/search over.

- For instance, for logistic regression, H is given by all classifiers of the form (parameterized by W,b):

$$h(X;W,b) = \frac{1}{1+\exp[-(W^T X + b)]}$$

Inductive Bias

- The assumption one makes to generalize beyond the training data.

- Examples:
  - 1-NN: the label is the same as that of the closest training example.

  - LL: the classification function is a smooth function of the form:

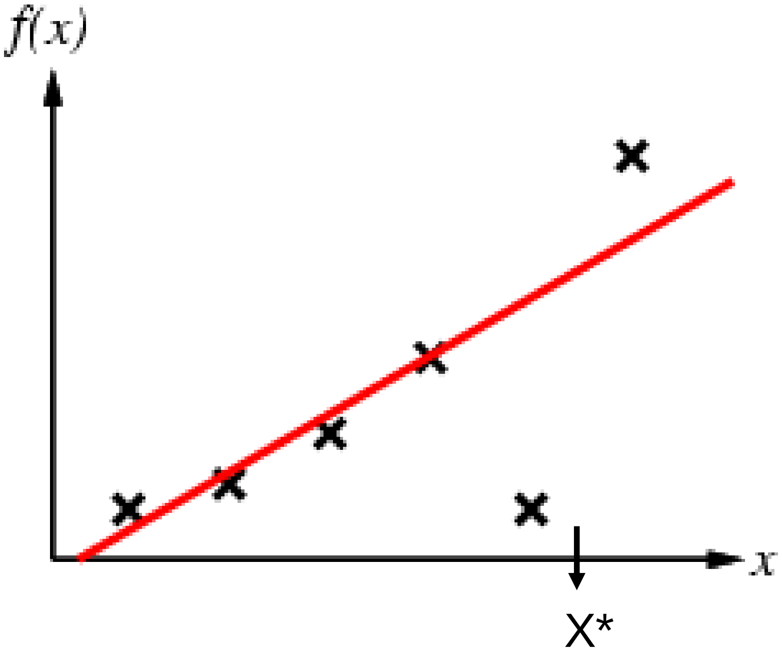
$$h(X;W,b) = \frac{1}{1+\exp[-(W^T X + b)]}$$

- Without inductive bias (i.e. without assumptions) there is no generalization possible! (you have not expressed preference for unseen data in any way).

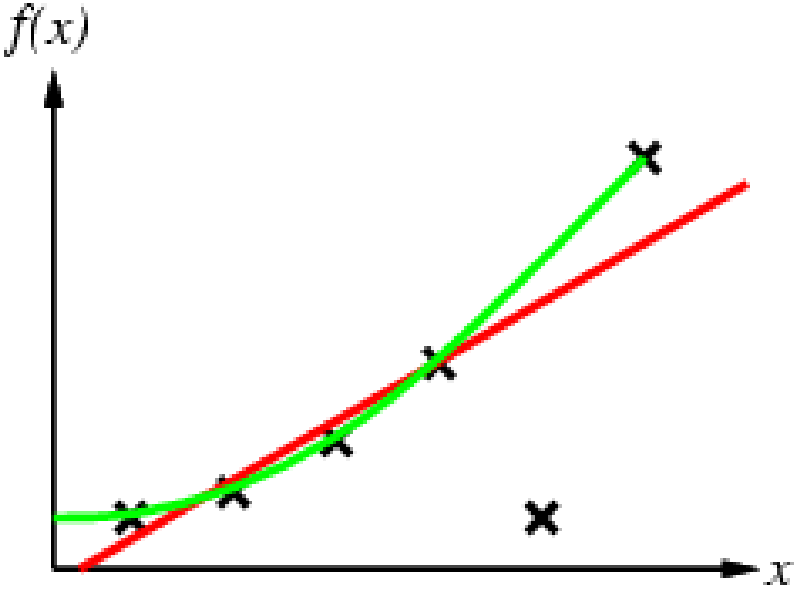
- Learning is hence converting your prior assumptions + the data into a classifier for new data.

Generalization

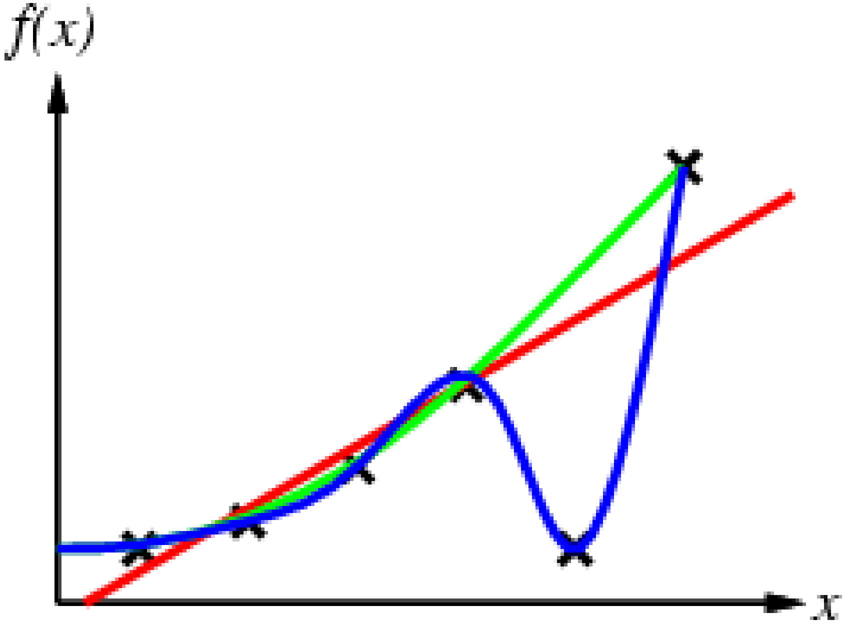
- Consider the following *regression* problem:
- Predict the real value on the y-axis from the real value on the x-axis.
- You are given 6 examples: {Xi,Yi}.
- What is the y-value for a new query point X* ?



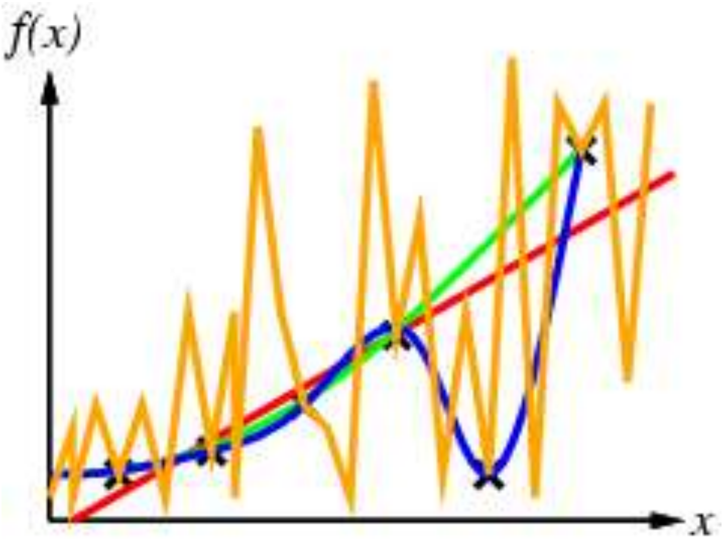
Generalization



Generalization

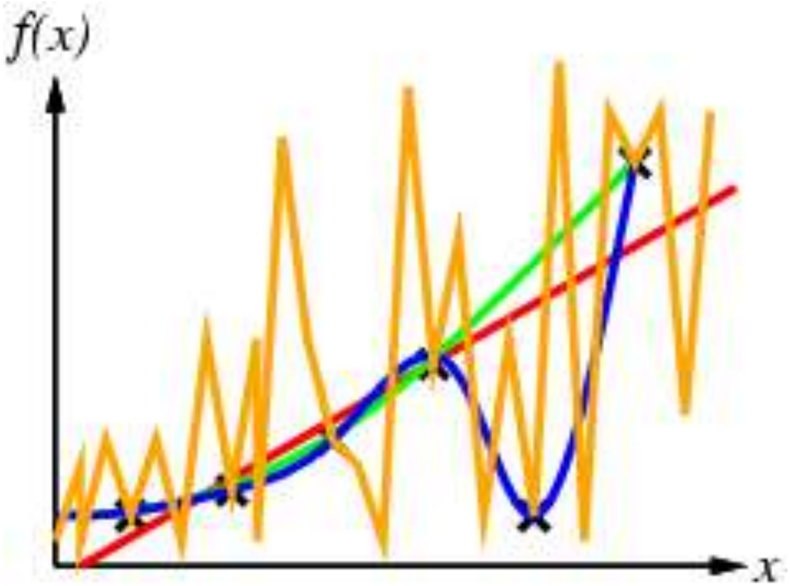


Generalization



which curve is best?

Generalization



- Ockham's razor: prefer the simplest hypothesis consistent with data.

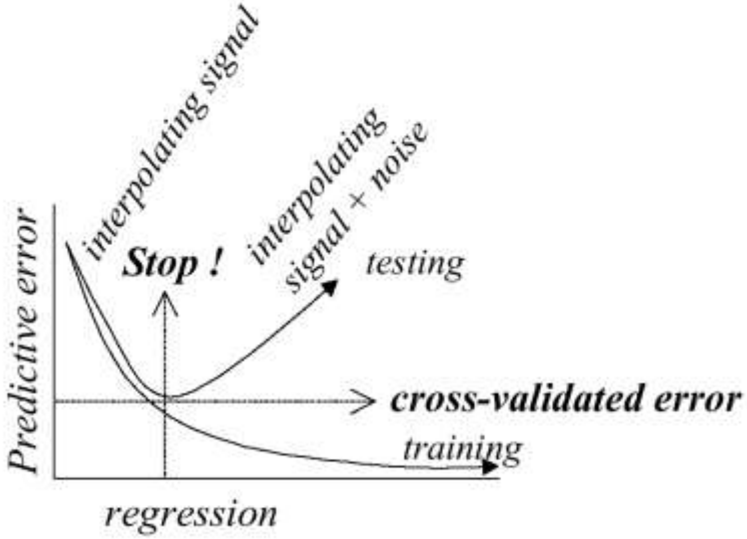
Generalization

Learning is concerned with accurate prediction
of future data, *not* accurate prediction of training data.

(The single most important sentence you will see in the course)

Cross-validation

How do we ensure good generalization, i.e. avoid "over-fitting" on our particular data sample.



- You are ultimately interested in good performance on new (unseen) test data.
- To estimate that, split off a (smallish) subset of the training data (called validation set).
- Train without validation data and "test" on validation data.
- Repeat this over multiple splits of the data and average results.
- Reasonable split: 90% train, 10% test, average over the 10 splits.

UNIT-II

by

An Overview of RS Image Clustering and Classification

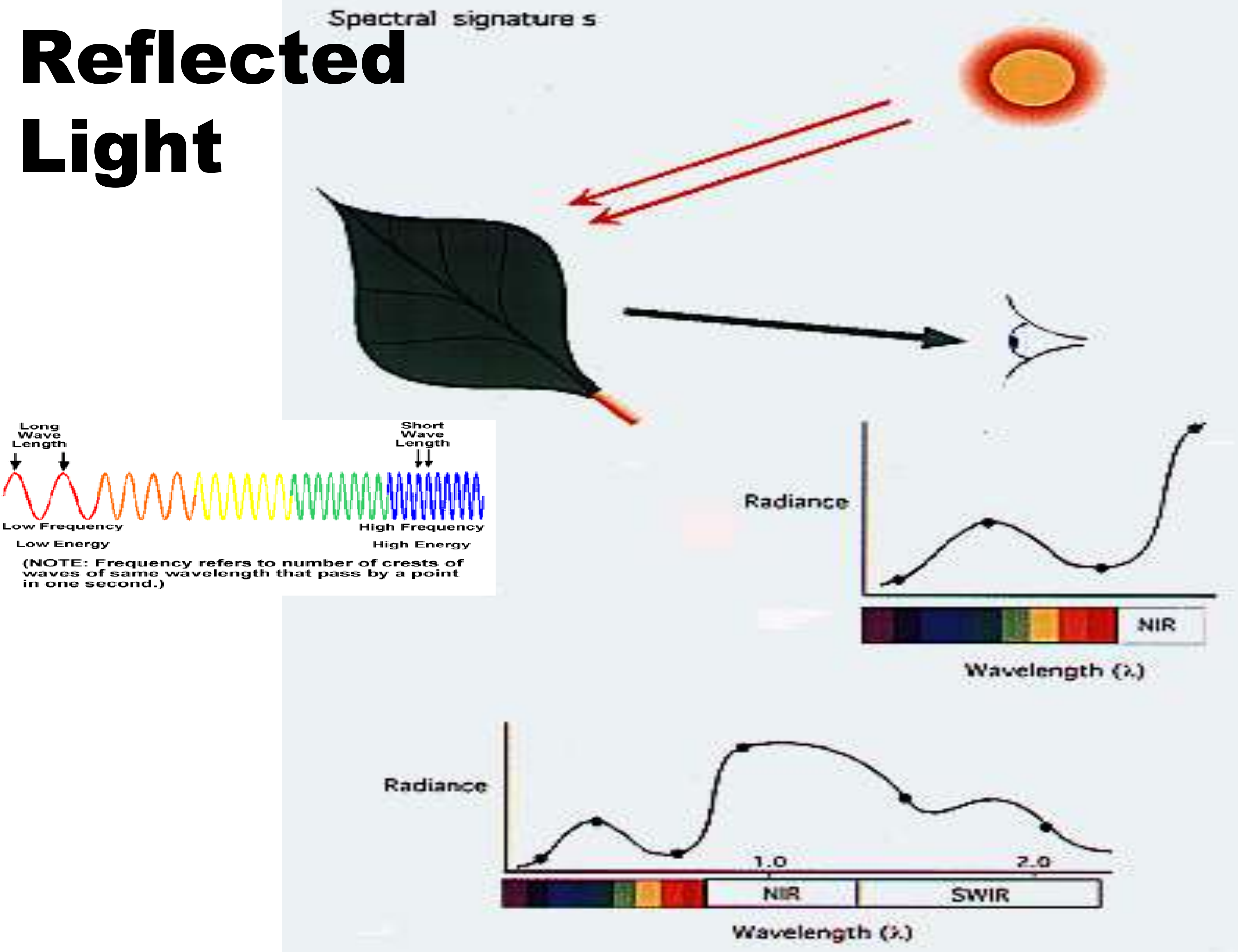
What is Remote Sensing and Image Classification?

- Remote Sensing is a technology for sampling radiation and force fields to acquire and interpret geospatial data to develop information about features, objects, and classes on Earth's land surface, oceans, and atmosphere (and, where applicable, on the exterior's of other bodies in the solar system).

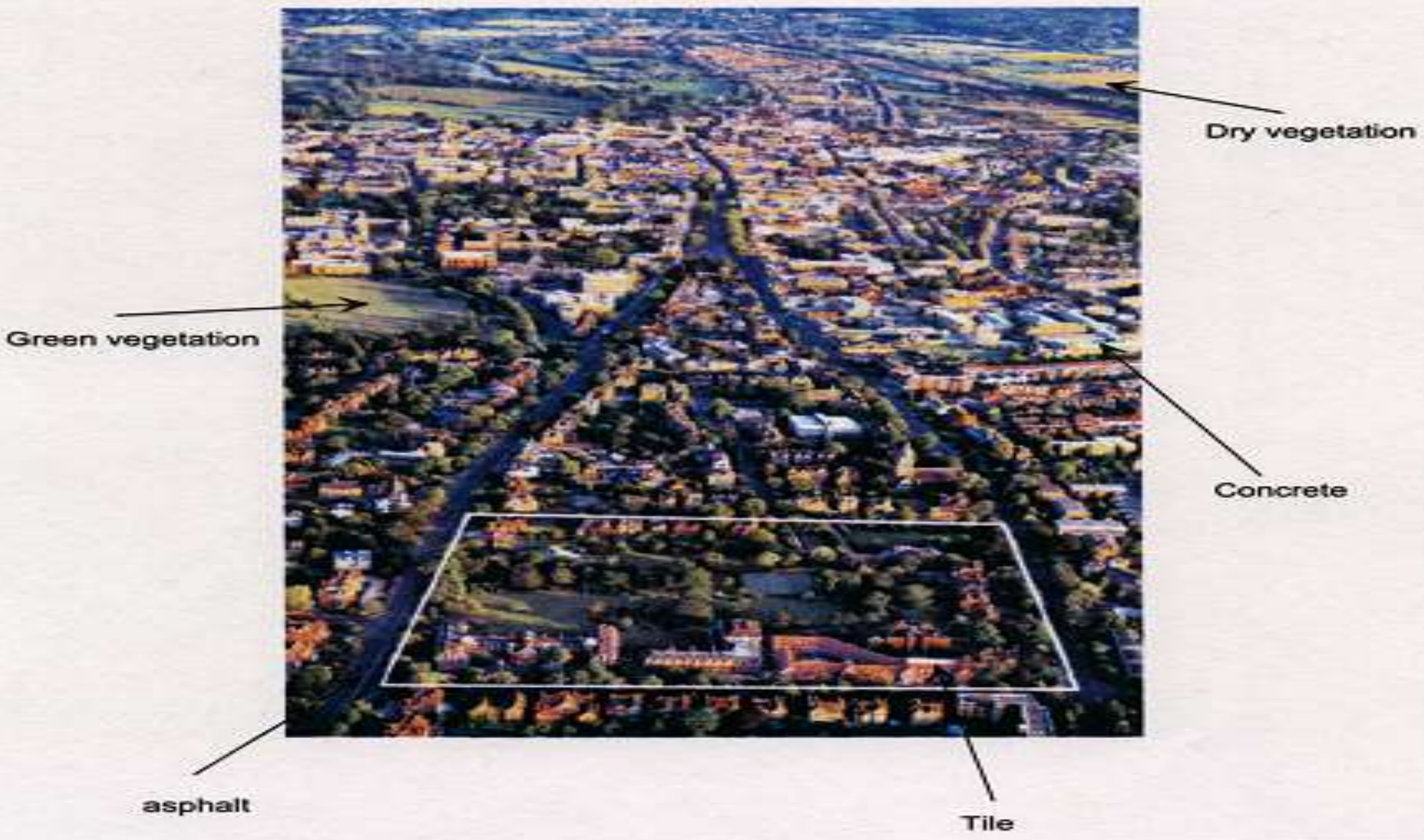
- Remote Sensing is detecting and measuring of electromagnetic energy (usually photons) emanating from distant objects made of various materials, so that we can identify and categorize these object by class or type, substance, and spatial distribution

- Image Classification has the overall objective to automatically categorize all pixels in an image into classes or themes. The Spectral pattern, or signature of surface materials belonging to a class or theme determines an assignment to a class.

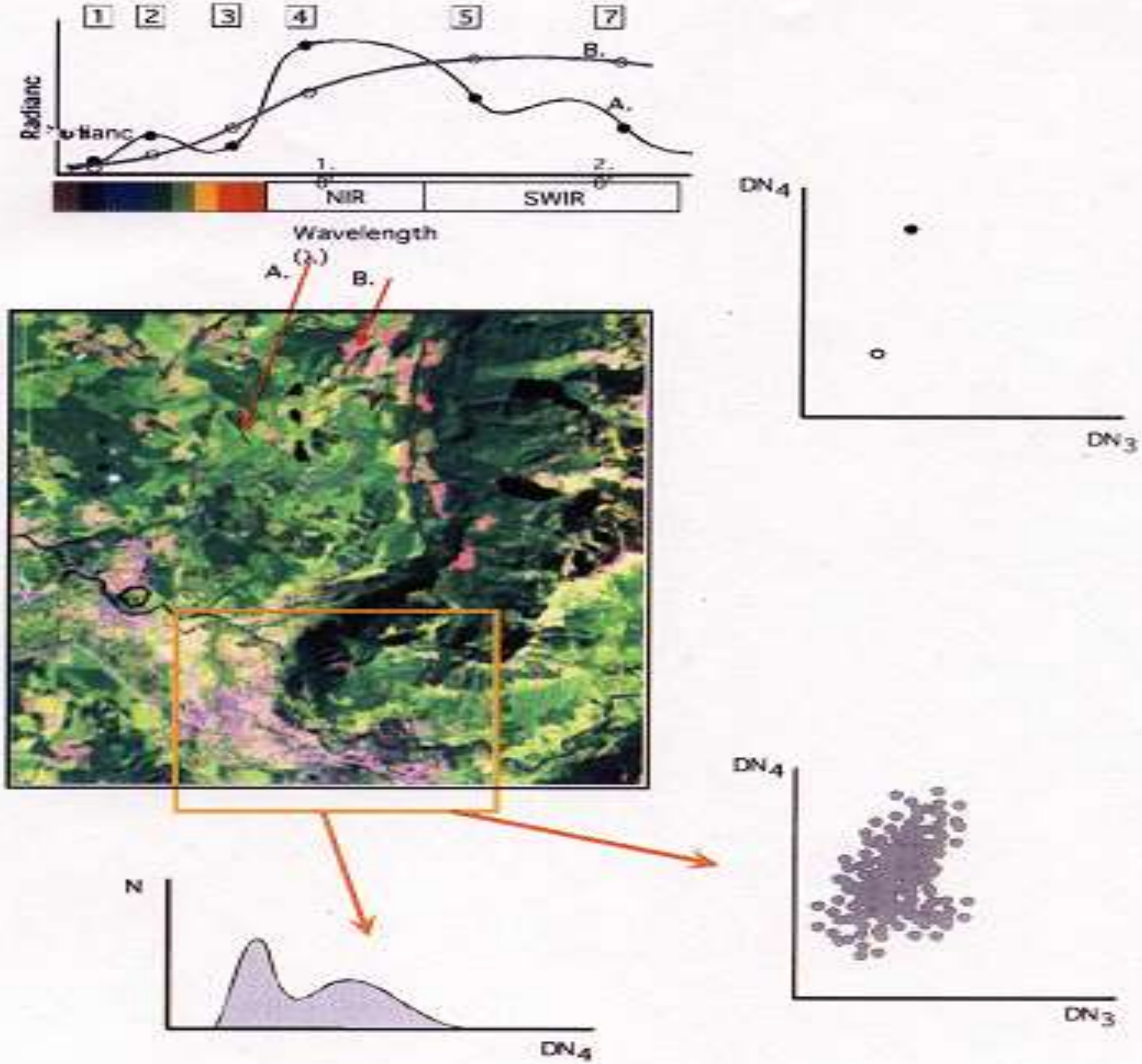
Reflected Light



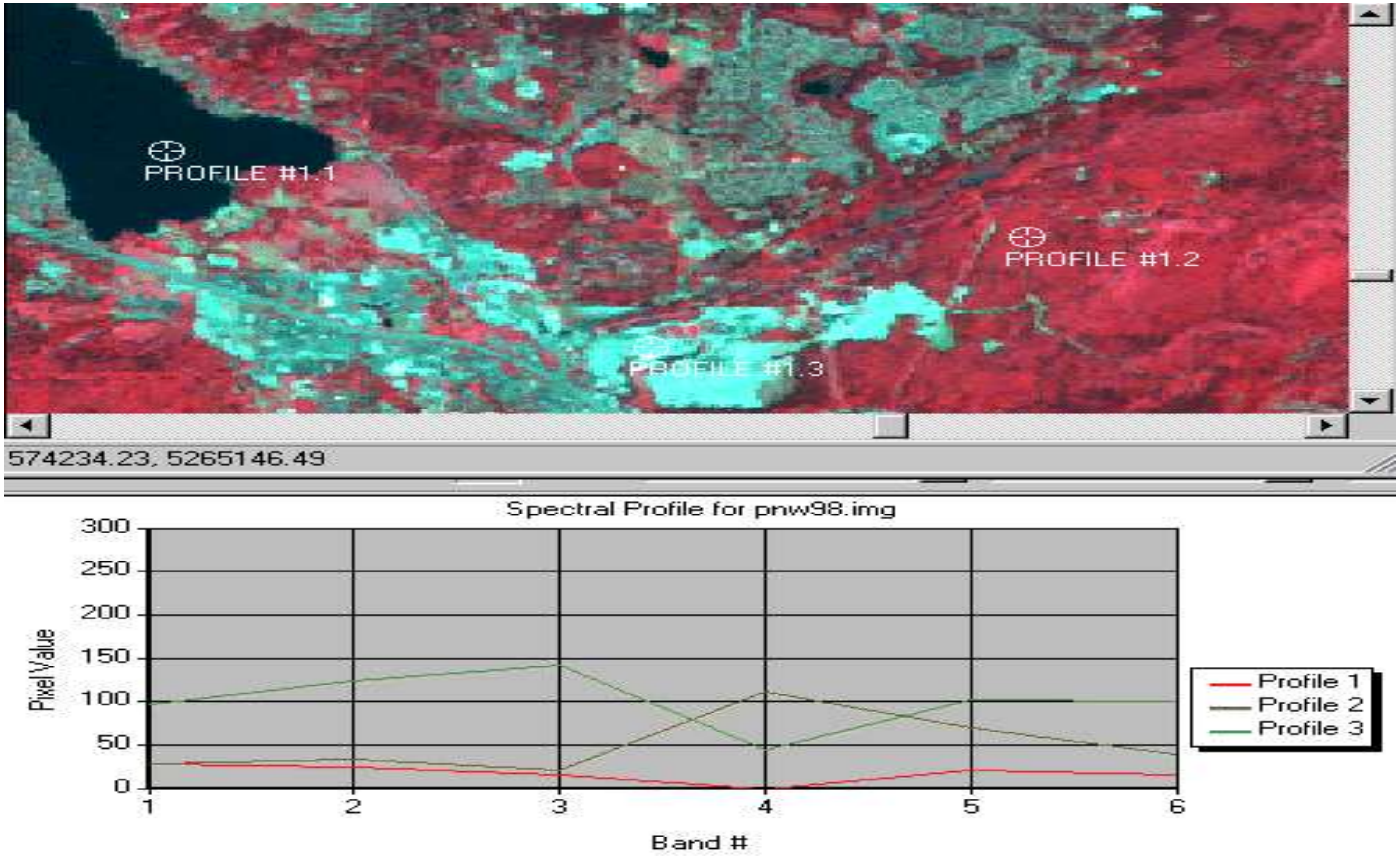
The “PIXEL”



Wavelength (Bands)

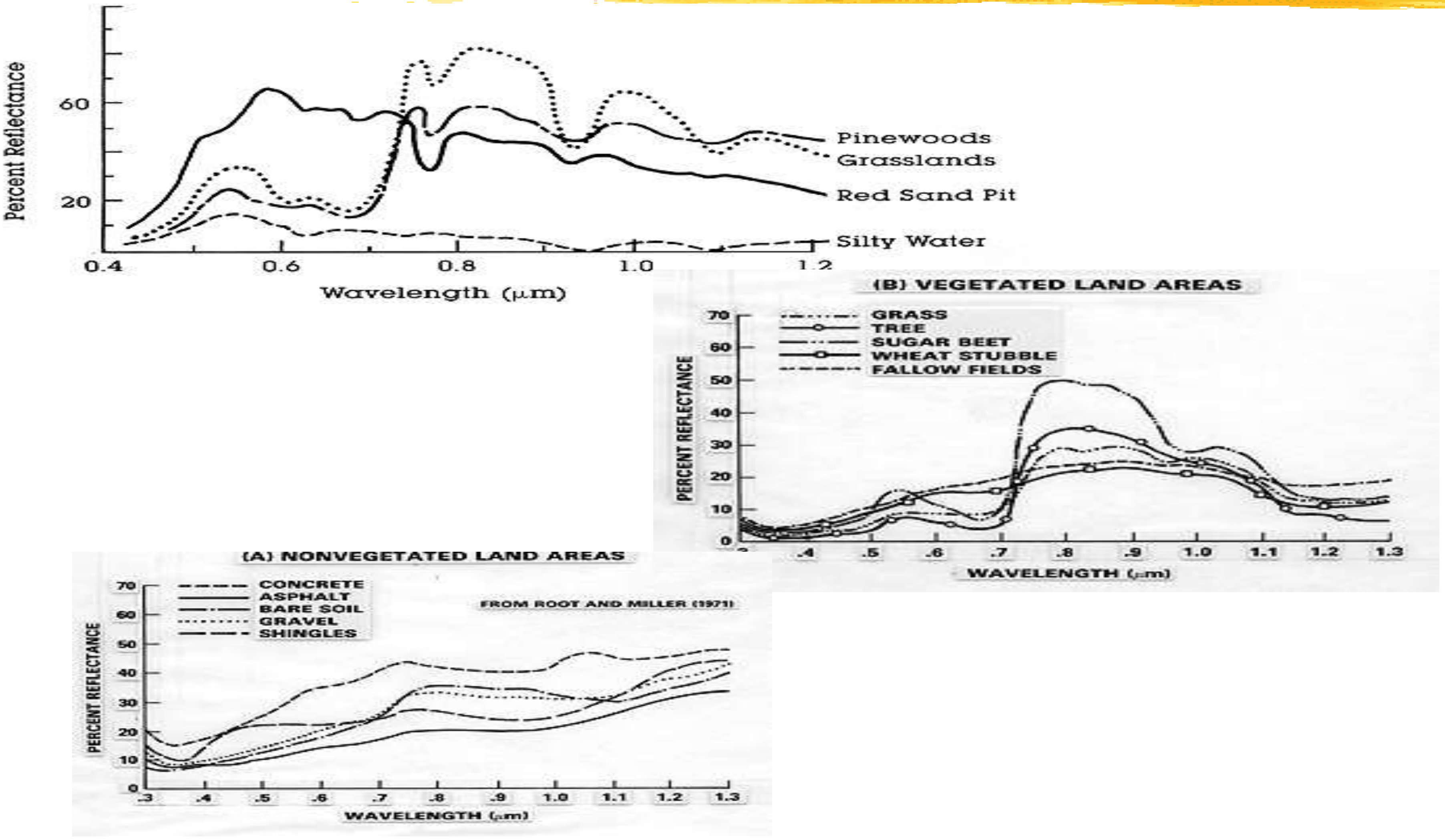


Spectral Profile

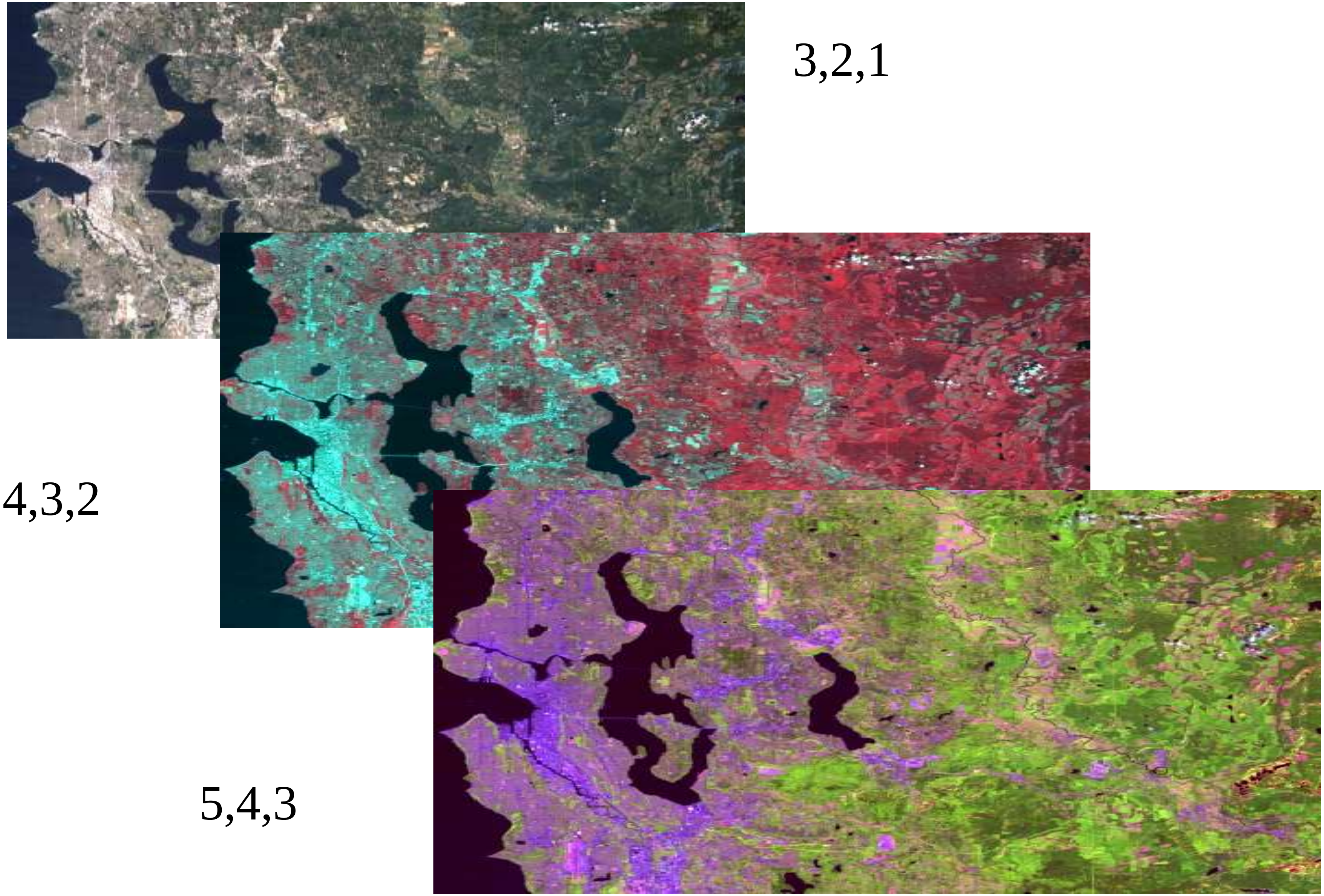


574234.23, 5265146.49

Spectral Signatures



Band Combinations

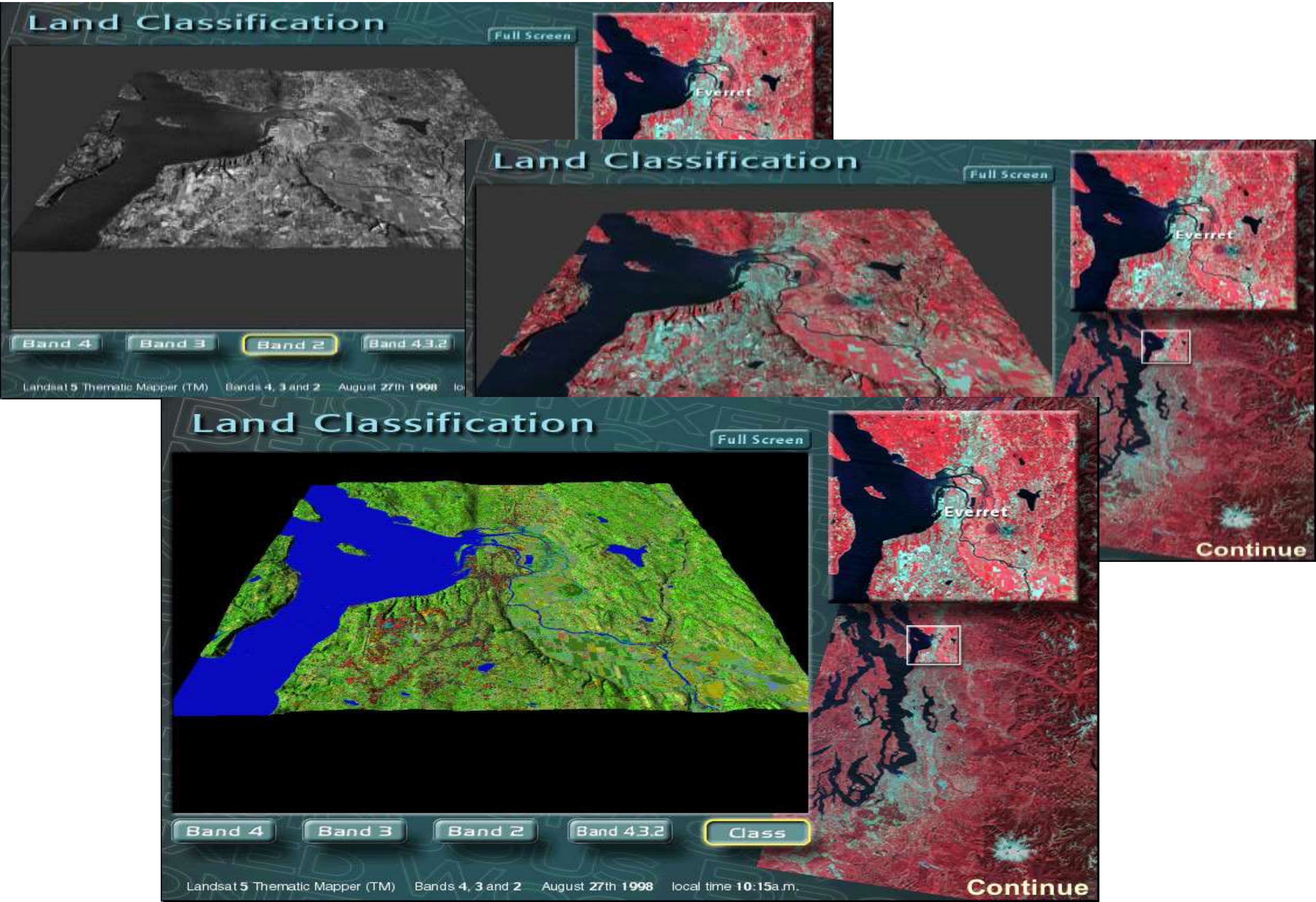


3,2,1

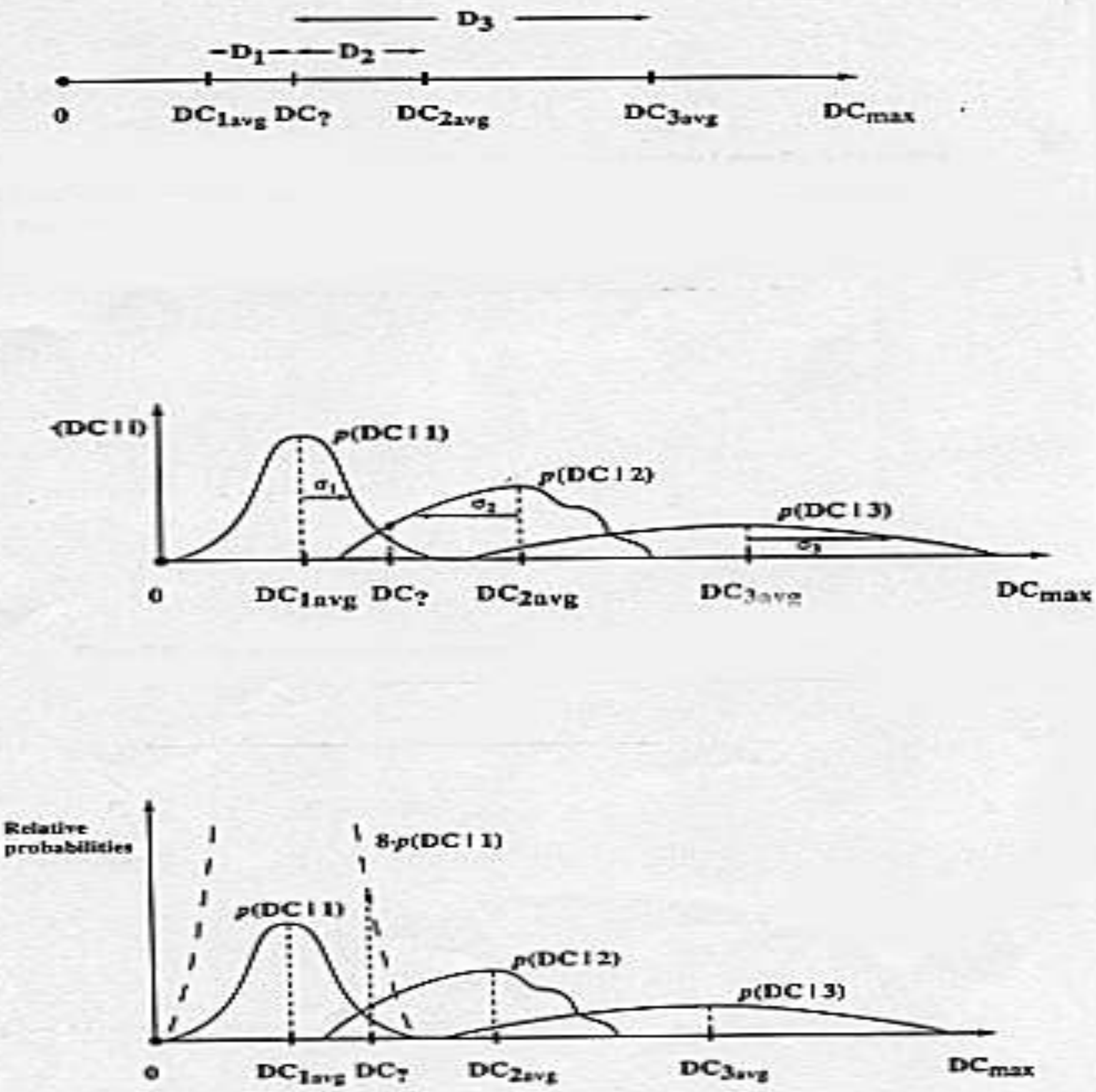
4,3,2

5,4,3

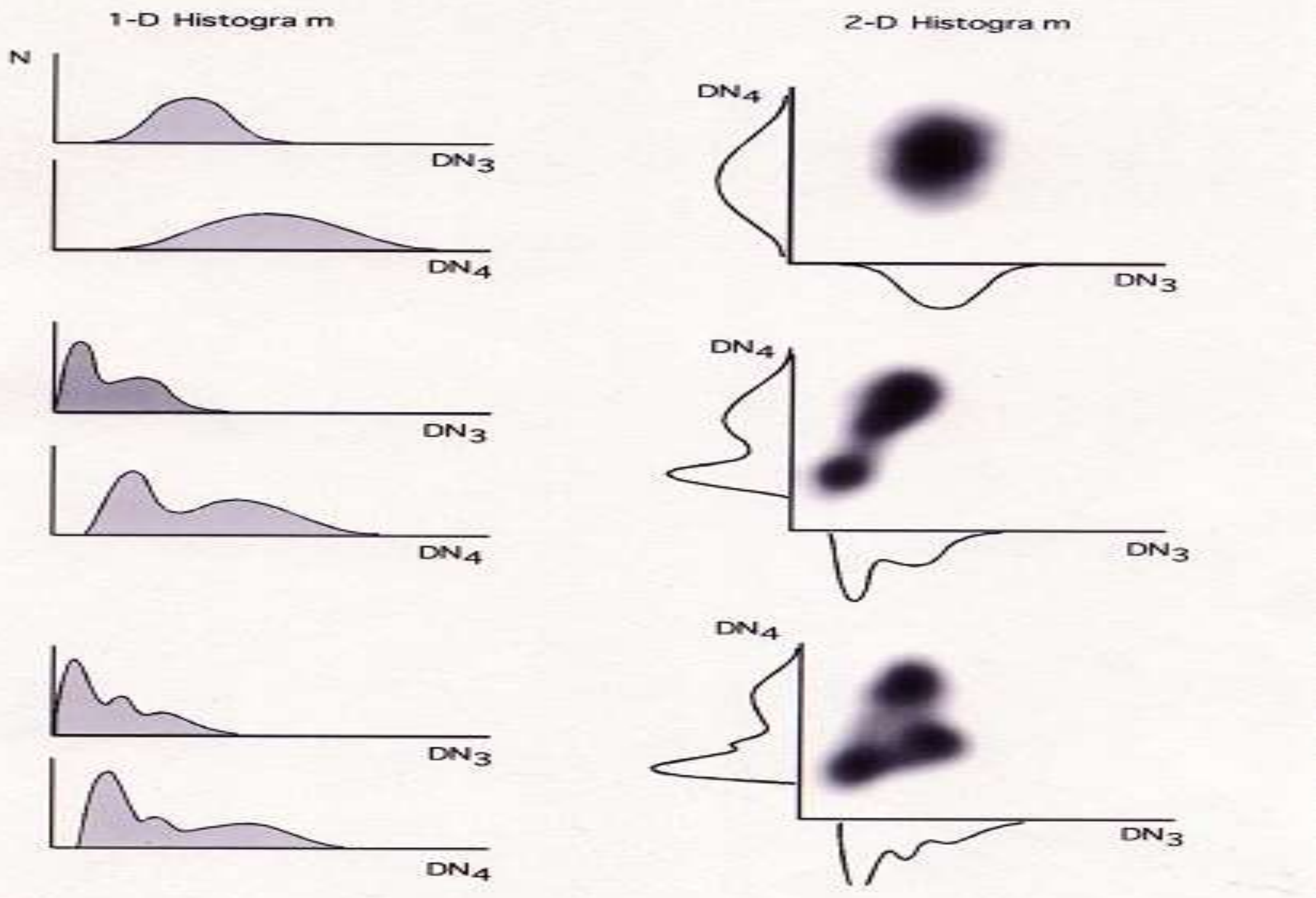
Image Classification



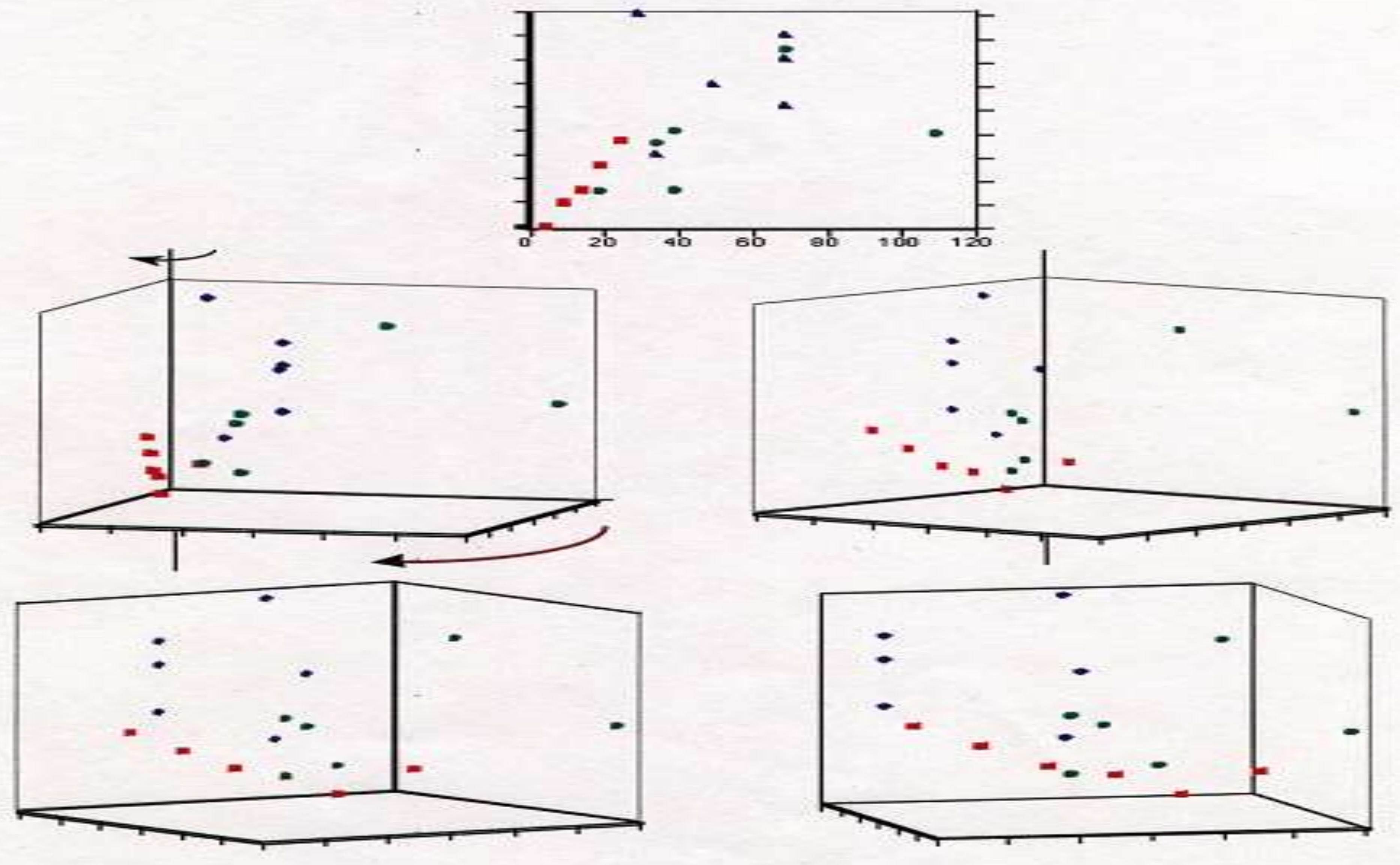
1d classifier



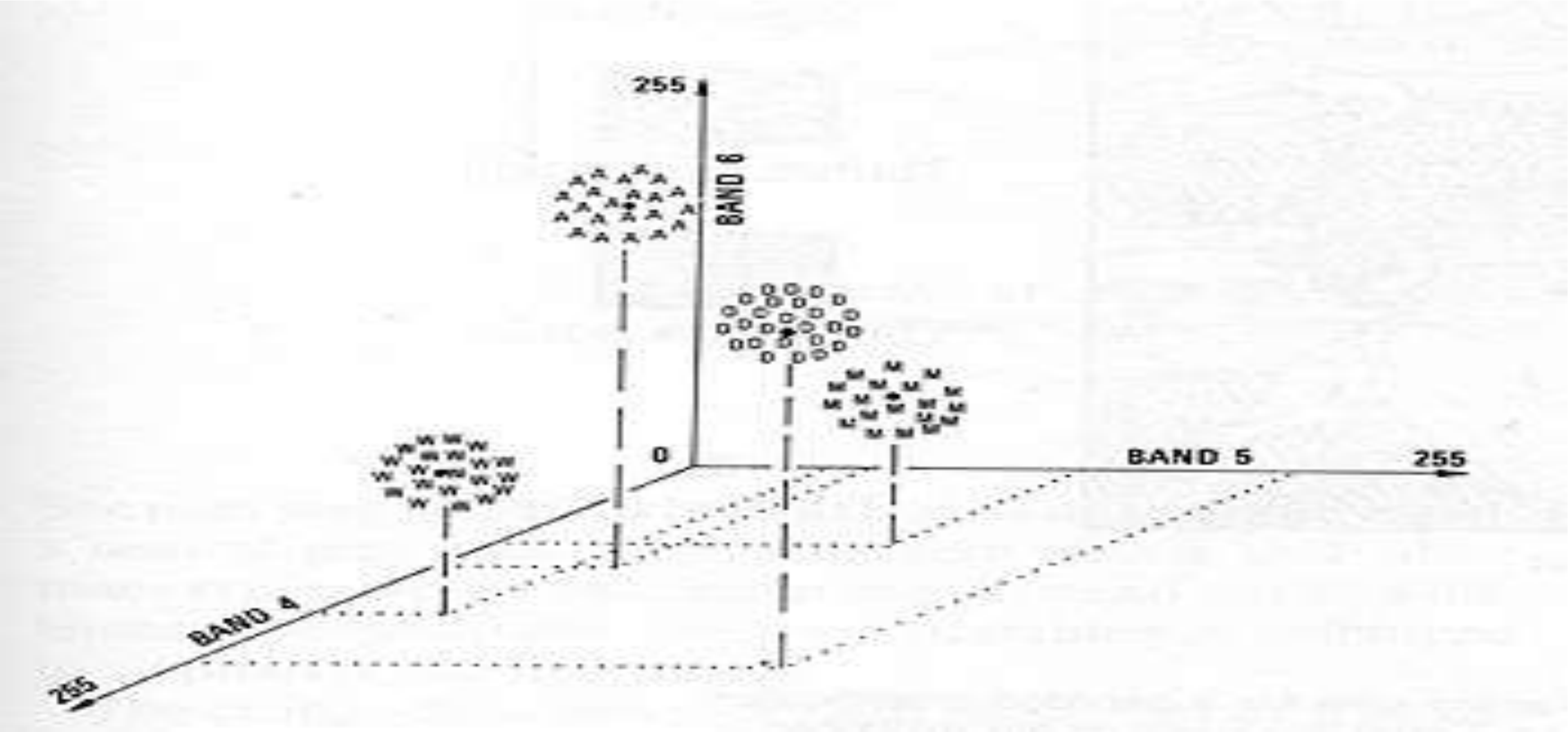
Spectral Dimensions



3 band space



Clusters



Dimensionality

N = the number of bands = dimensions

…. an (n) dimensional data (feature) space

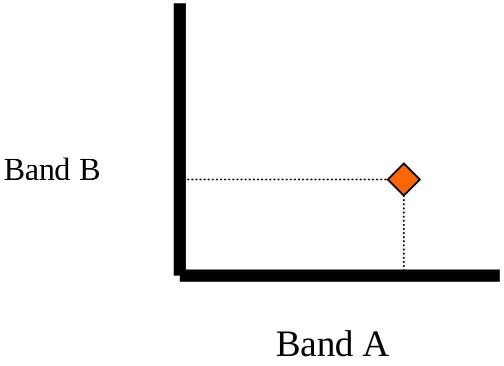
Measurement Vector

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}$$

Mean Vector

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_n \end{bmatrix}$$

Feature Space - 2dimensions



$$\begin{pmatrix} 190 \\ 85 \end{pmatrix}$$

Spectral Distance

* a number that allows two measurement vectors to be compared

$$D = \sqrt{\sum_{i=1}^{n} \left(d_i - e_i\right)^2}$$

$i = a$ band (dimension)

$\mathrm{d_i}$ = value of pixel d in band i

e_i = value of pixel e in band i

Classification Approaches

- Unsupervised: self organizing
- Supervised: training
- Hybrid: self organization by categories
- Spectral Mixture Analysis: sub-pixel variations.

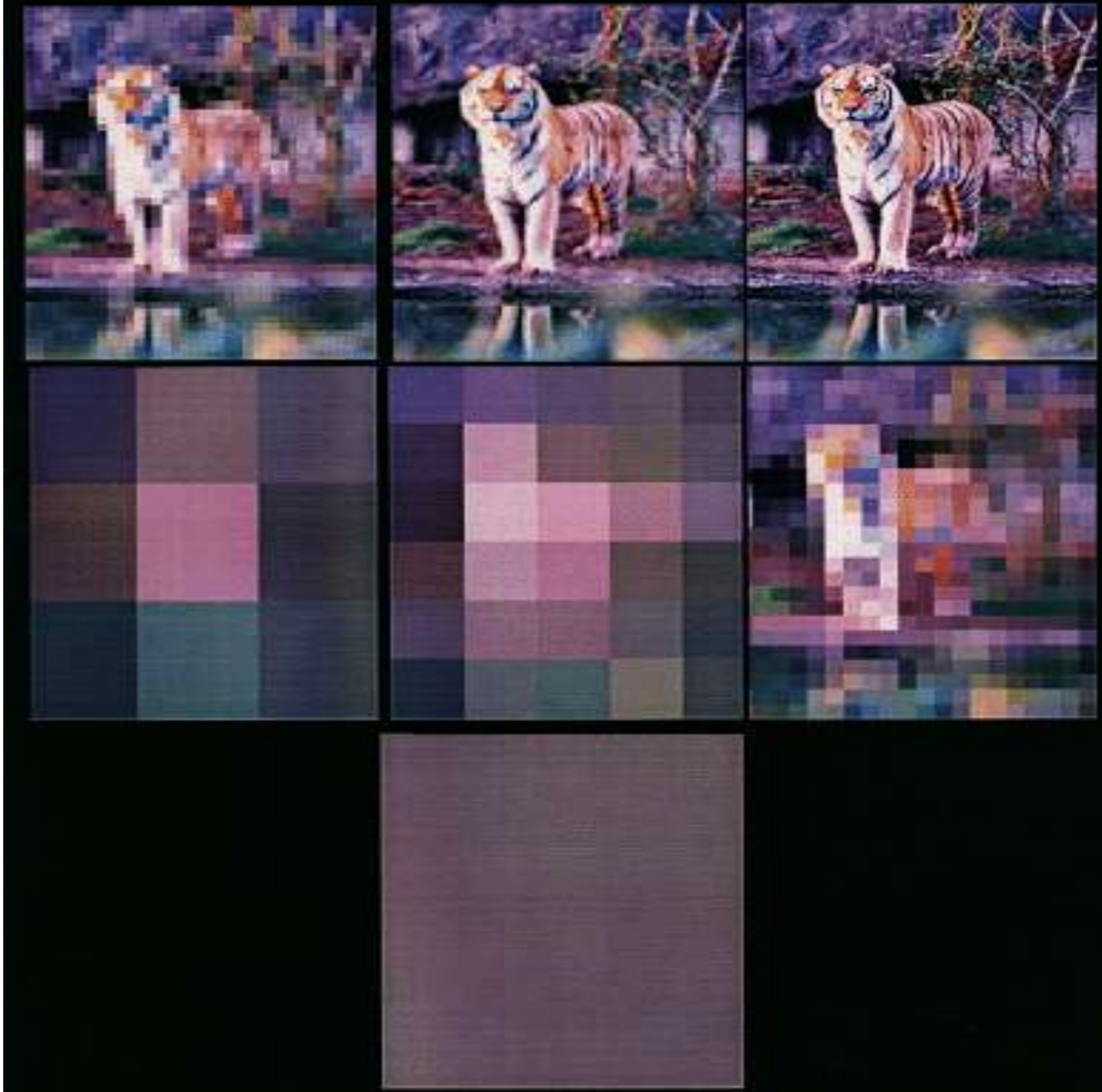
Clustering / Classification

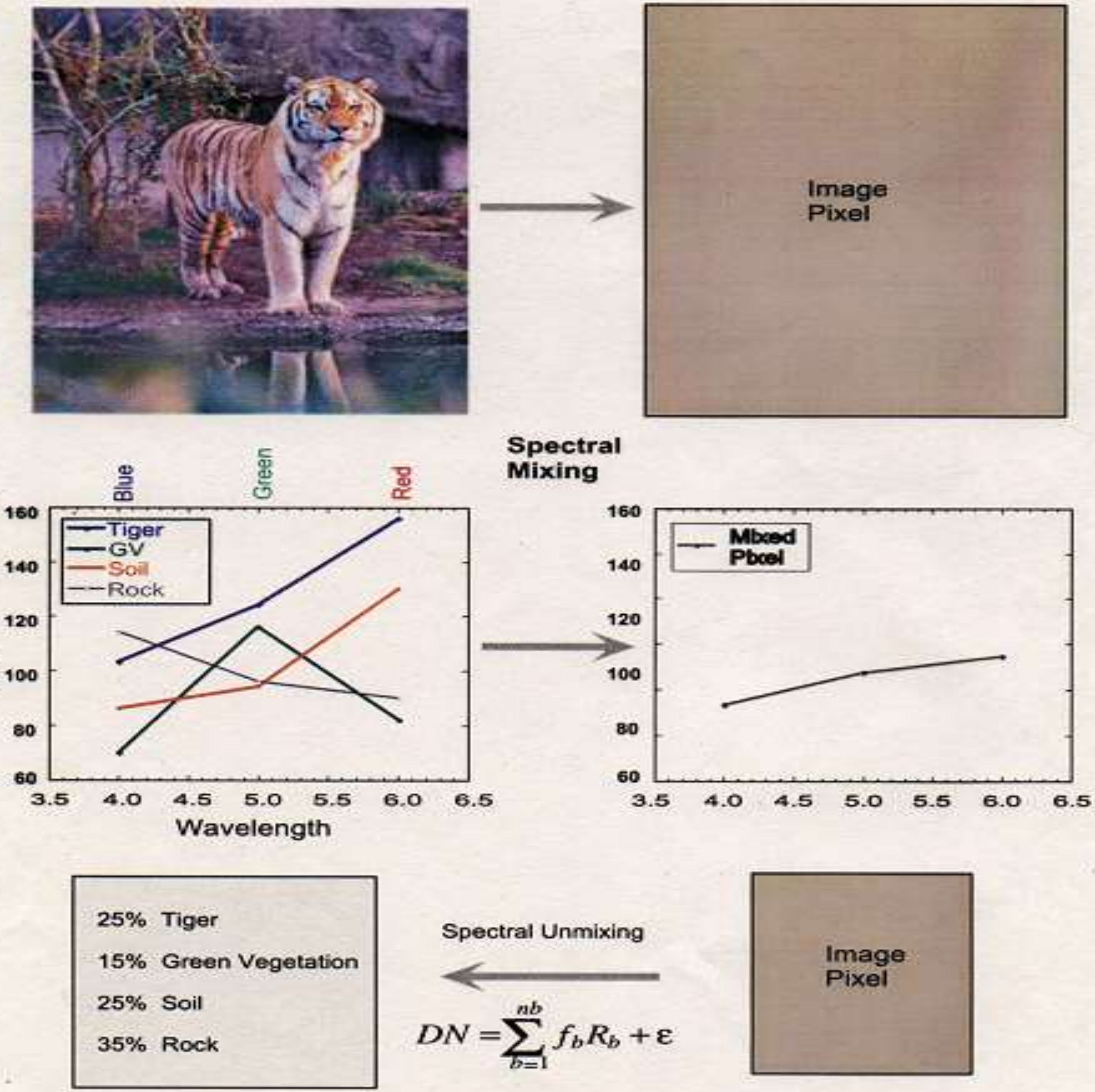
- Clustering or Training Stage:
  - Through actions of either the analyst's supervision or an unsupervised algorithm, a numeric description of the spectral attribute of each "class" is determined (a multi-spectral cluster mean signature).
- Classification Stage:
  - By comparing the spectral signature to of a pixel (the measure signature) to the each cluster signature a pixel is assigned to a category or class.

terms

- Parametric = based upon statistical parameters (mean & standard deviation)
- Non-Parametric = based upon objects (polygons) in feature space
- Decision Rules = rules for sorting pixels into classes

Resolution and Spectral Mixing





Spectral Mixing

25% Tiger

15% Green Vegetation

25% Soil

35% Rock

Spectral Unmixing

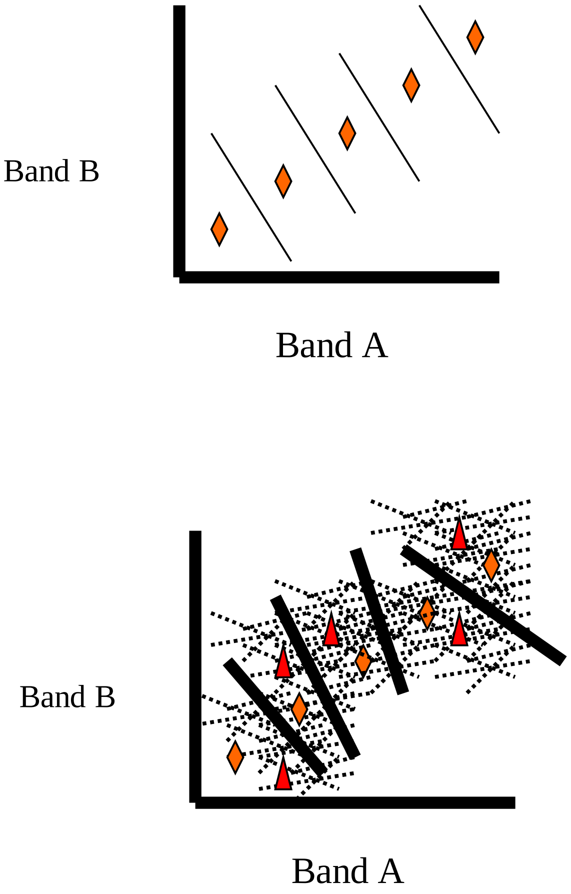
$$DN = \sum_{b=1}^{nb} f_b R_b + \varepsilon$$

Clustering

Minimum Spectral Distance - unsupervised

ISODATA

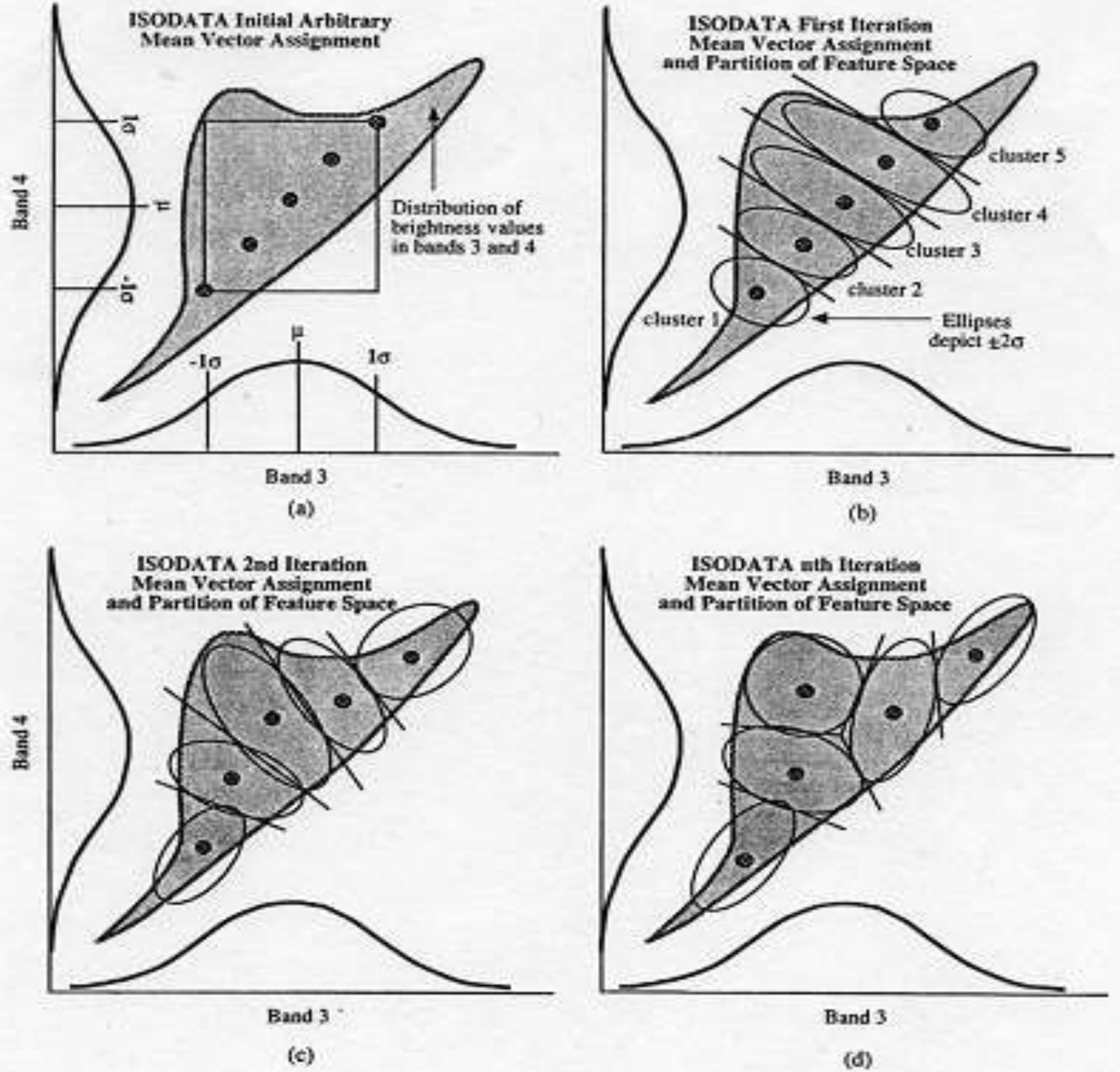
I - iterative
S - self
O - organizing
D - data
A - analysis
T - technique
A - (application)?



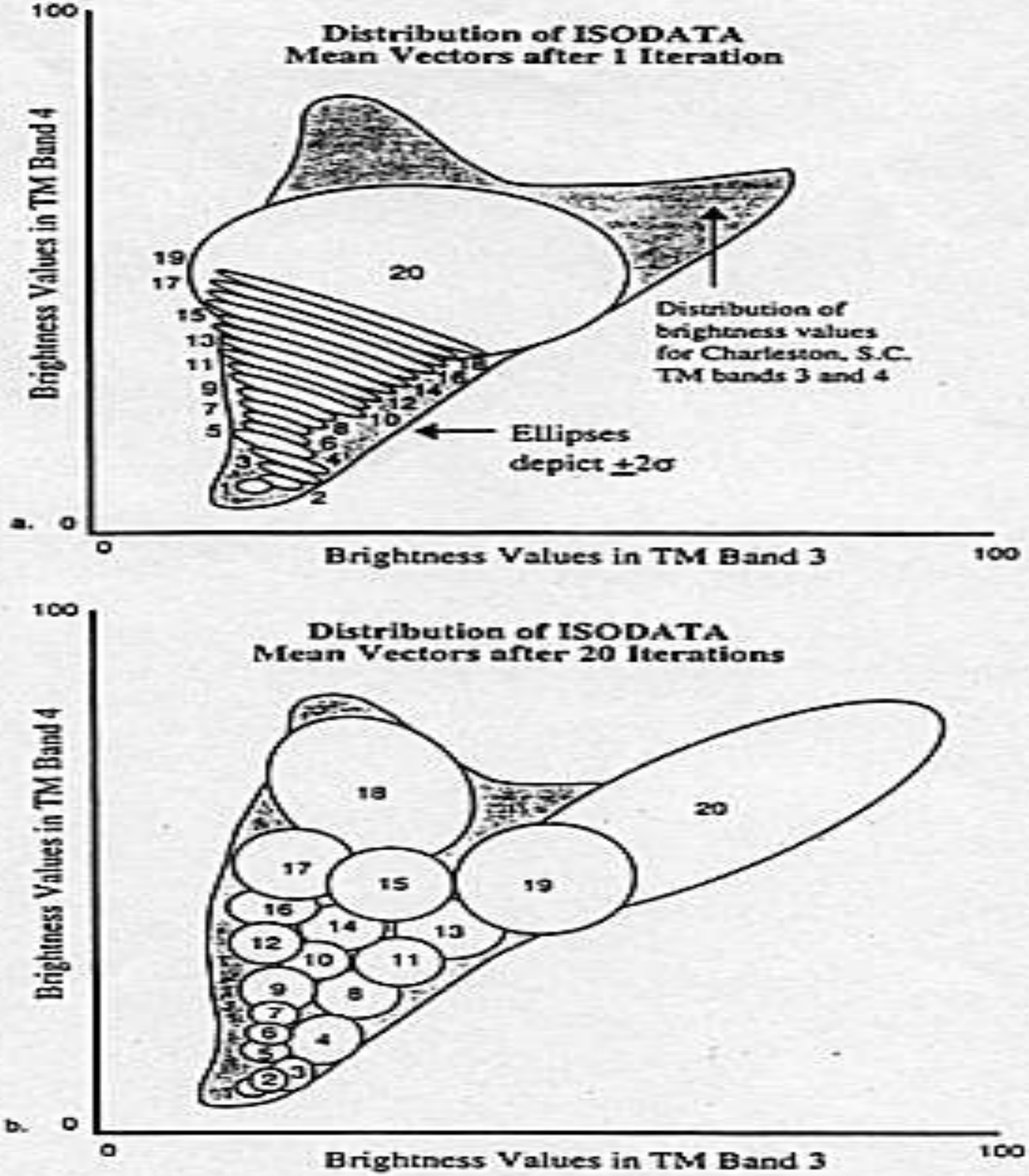
◆ 1st iteration cluster mean

▲ 2nd iteration cluster mean

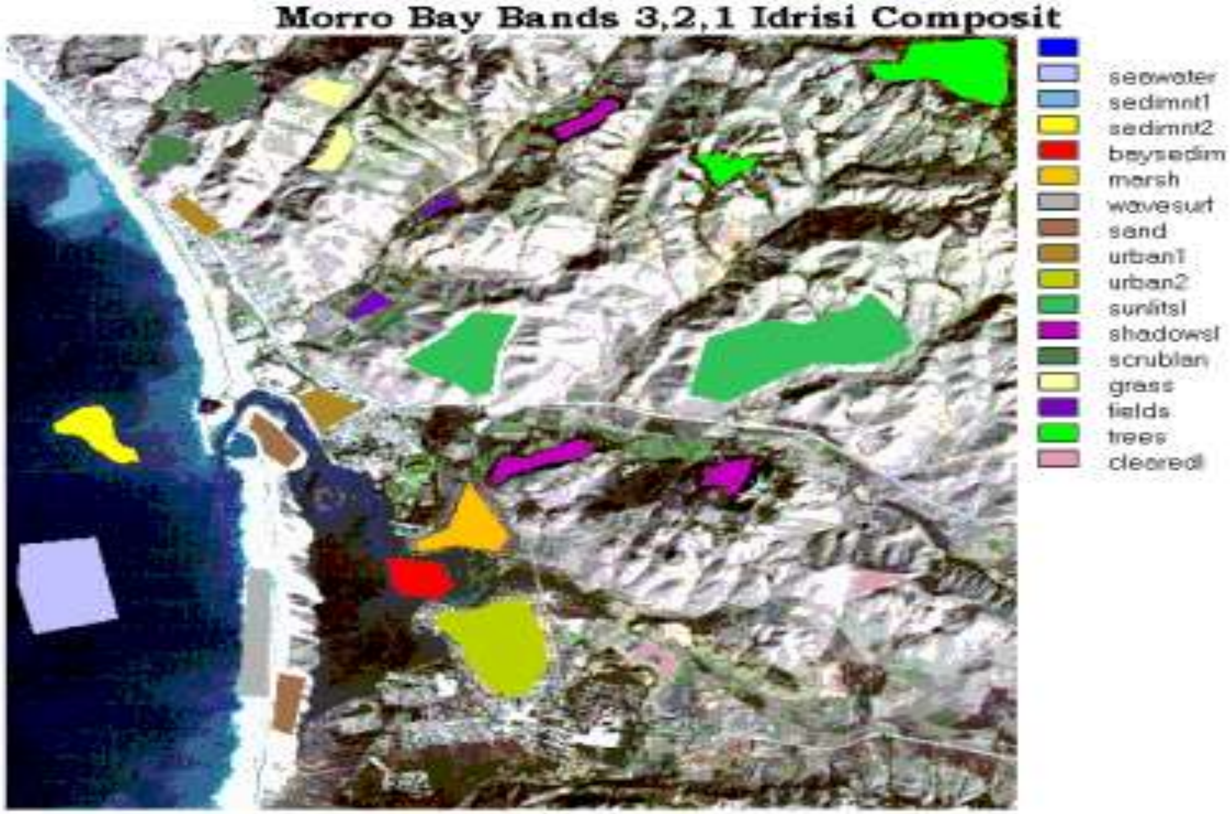
ISODATA clusters

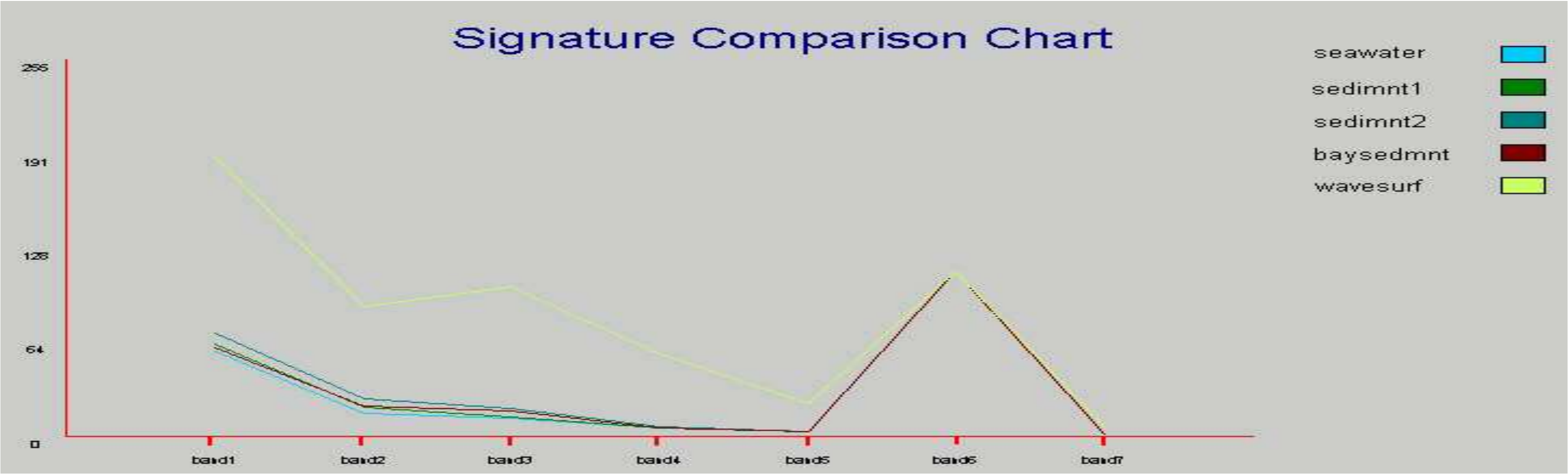


Unsupervised Classification ISODATA - Iterative Self-Organizing Data Analysis Technique



Supervised Classification





Classification Decision Rules

- If the non-parametric test results in one unique class, the pixel will be assigned to that class.
- if the non-parametric test results in zero classes (outside the decision boundaries) the the “unclassified rule applies … either left unclassified or classified by the parametric rule
- if the pixel falls into more than one class the overlap rule applies … left unclassified, use the parametric rule, or processing order

Non-Parametric

- parallelepiped
- feature space

Unclassified Options

- parametric rule
- unclassified

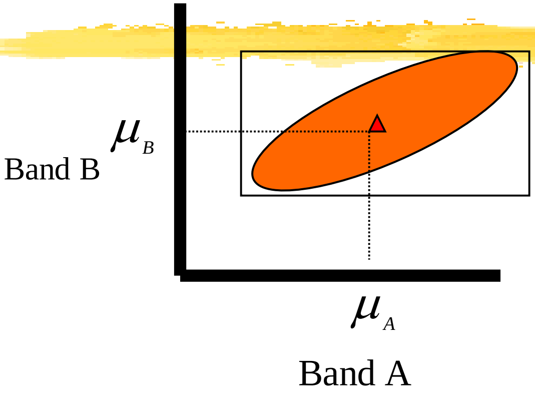
Overlap Options

- parametric rule
- by order
- unclassified

Parametric

- minimum distance
- Mahalanobis distance
- maximum likelihood

Parallelepiped



Maximum likelihood

(bayesian)

- •probability
- •Bayesian, a prior (weights)

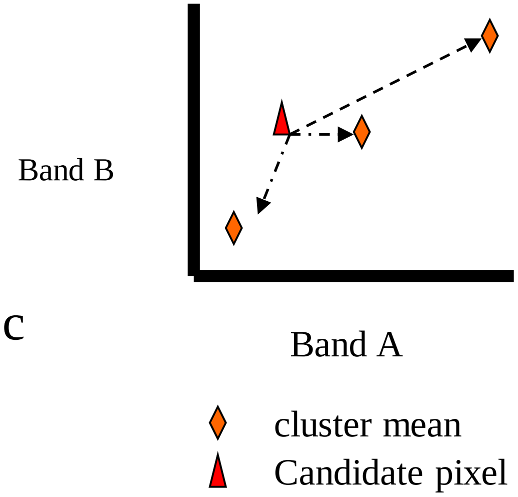
Minimum Distance

$$SD_{xyc} = \sqrt{\sum_{i=1}^{n}\left(\mu_{ci} - X_{xyi}\right)^2}$$

c = class

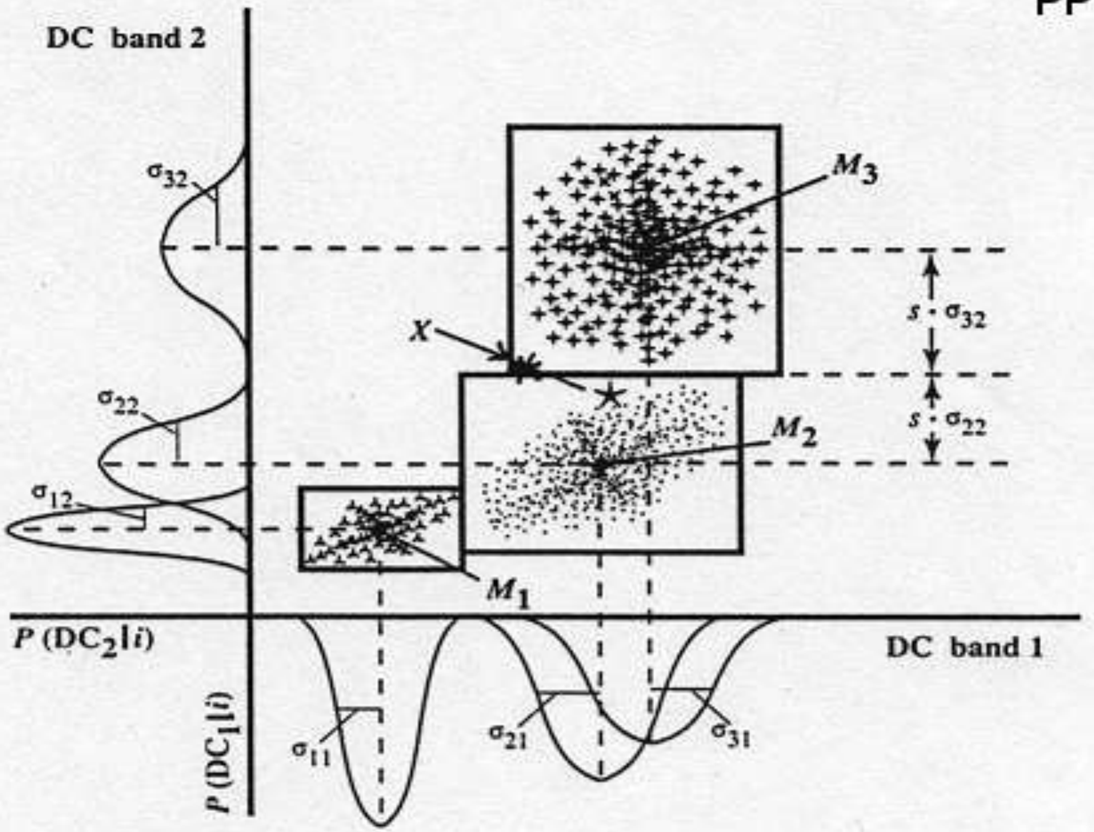
X_{xyi} = value of pixel x, y in i class

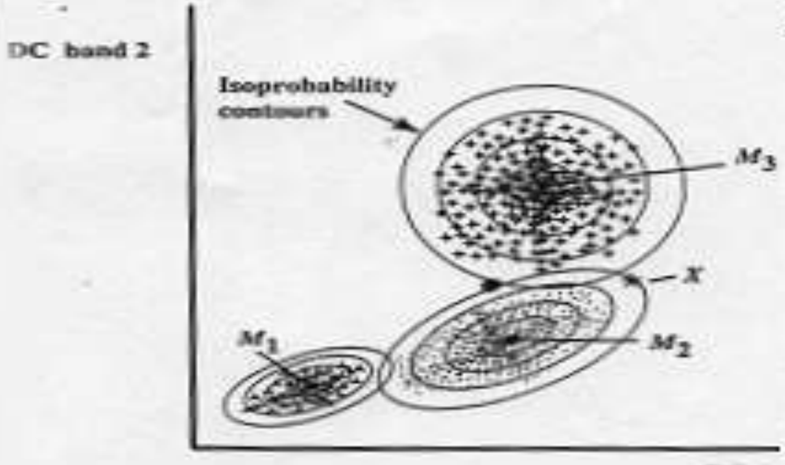
μ_{ci} = mean of values in i for sample for class c

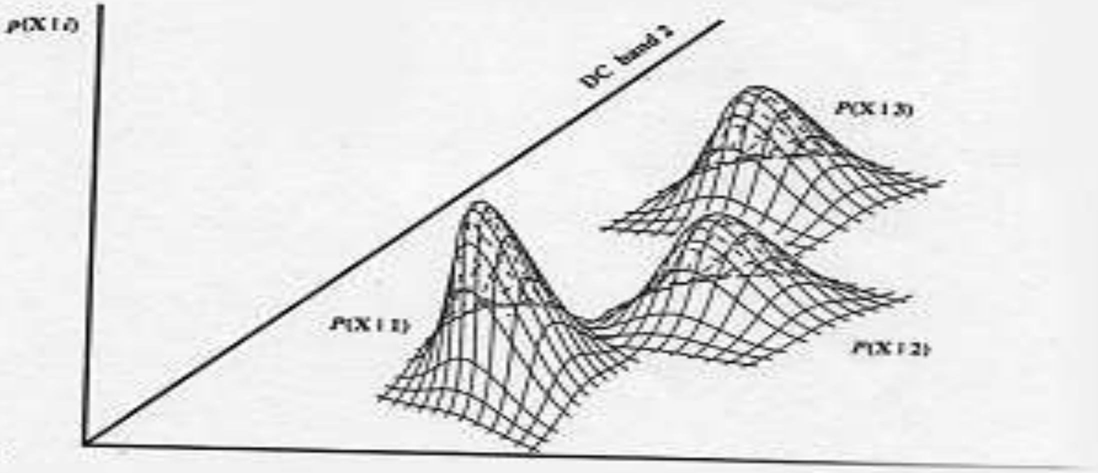


PP

Parametric classifiers







Classification Systems

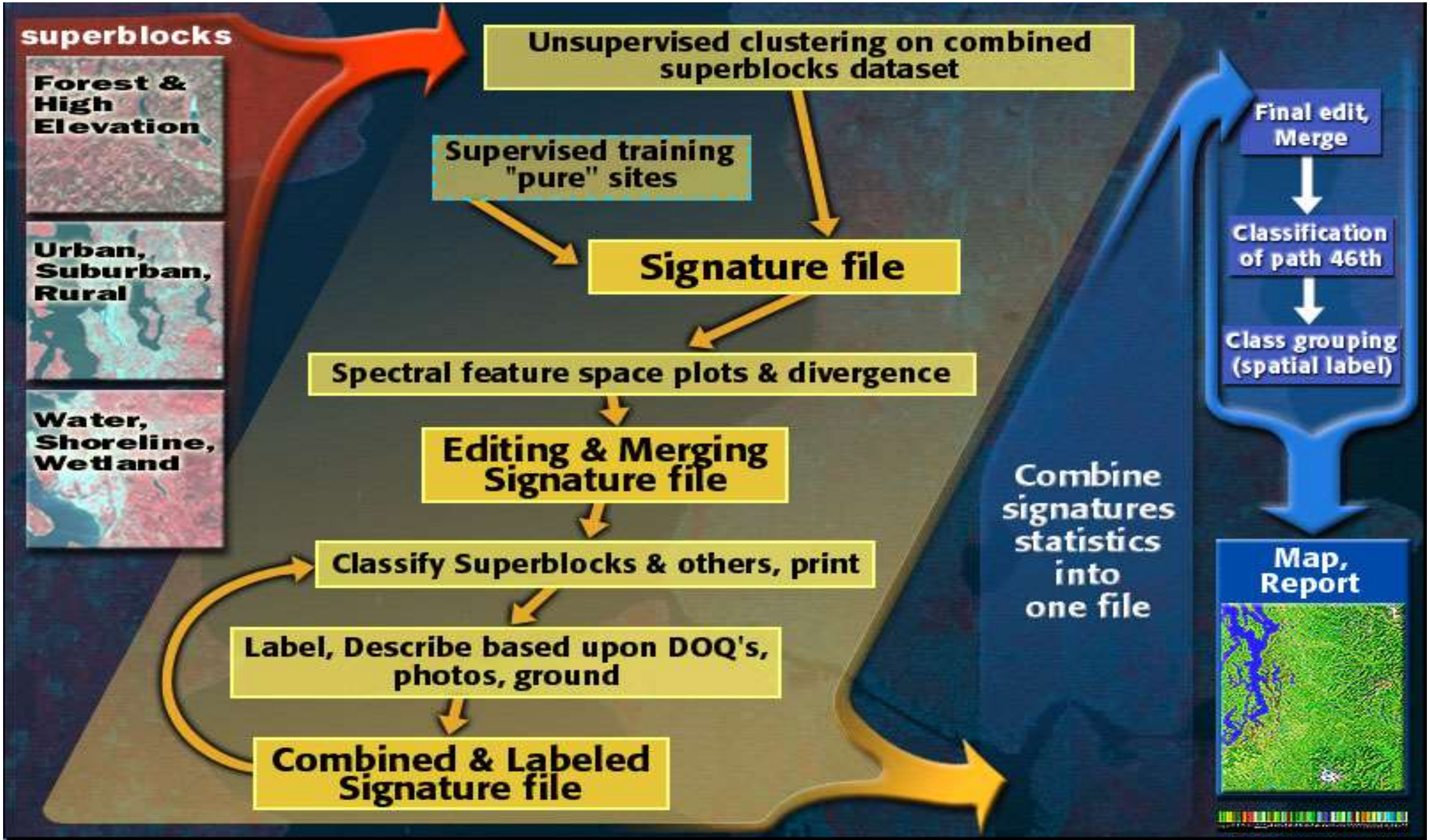
USGS - U.S. Geological Survey Land Cover Classification Scheme for Remote Sensor Data
USFW - U.S. Fish & Wildlife Wetland Classification System
NOAA CCAP - C-CAP Landcover Classification System, and Definitions
NOAA CCAP - C-CAP Wetland Classification Scheme Definitions
PRISM - PRISM General Landcover
King Co. - King County General Landcover (specific use, by Chris Pyle)

Level

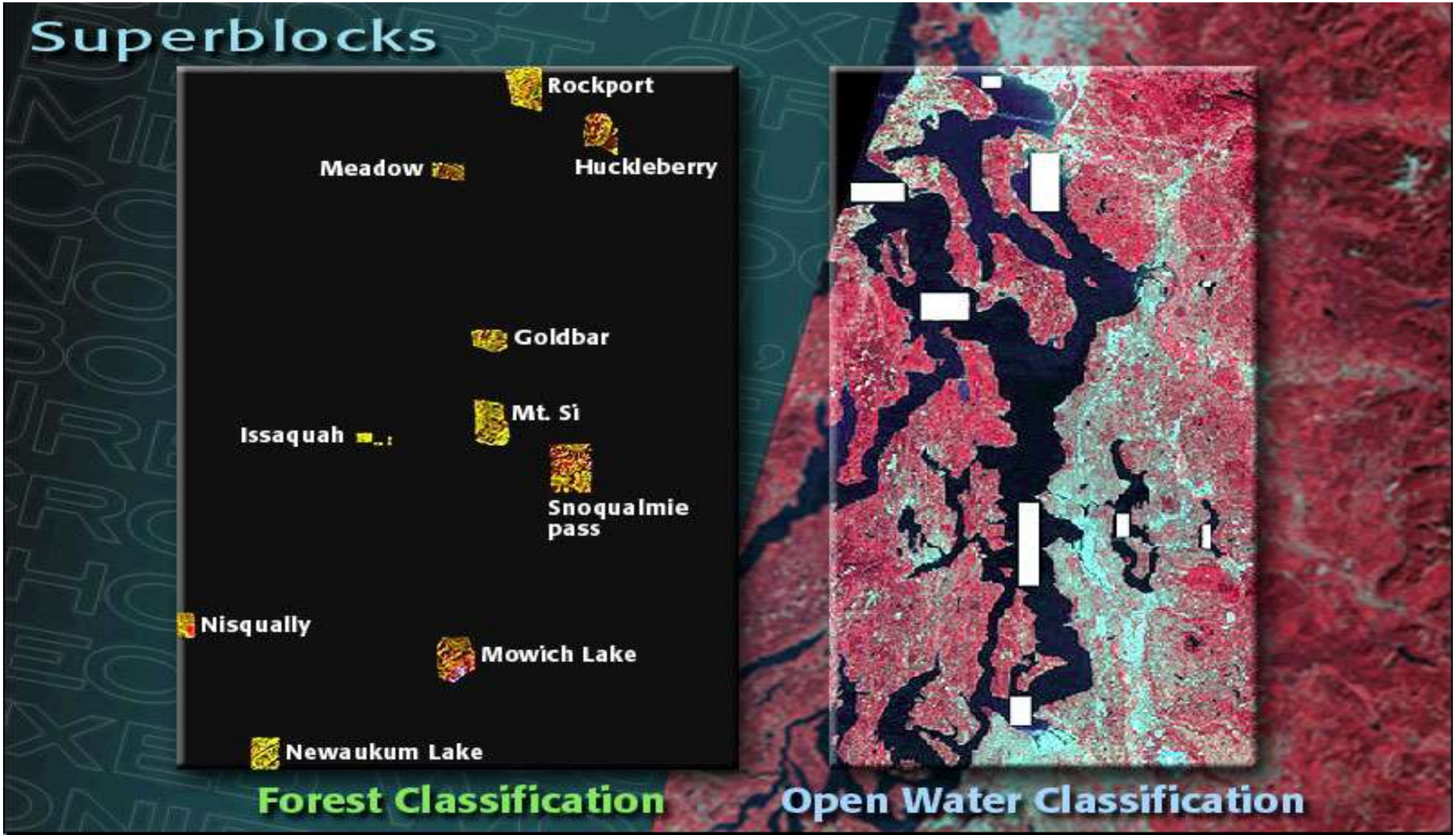
- 1 Urban or Built-Up Land
  - 11 Residential
  - 12 Commercial and Services
  - 13 Industrial
  - 14 Transportation, Communications and Utilities
  - 15 Industrial and Commercial Complexes
  - 16 Mixed Urban or Built-Up
  - 17 Other Urban or Built-up Land
- 2 Agricultural Land
  - 21 Cropland and Pasture
  - 22 Orchards, Groves, Vineyards, Nurseries and Ornamental Horticultural Areas
  - 23 Confined Feeding Operations
  - 24 Other Agricultural Land

http://boto.ocean.washington.edu/oc_gis_rs/lawrs/classify.html

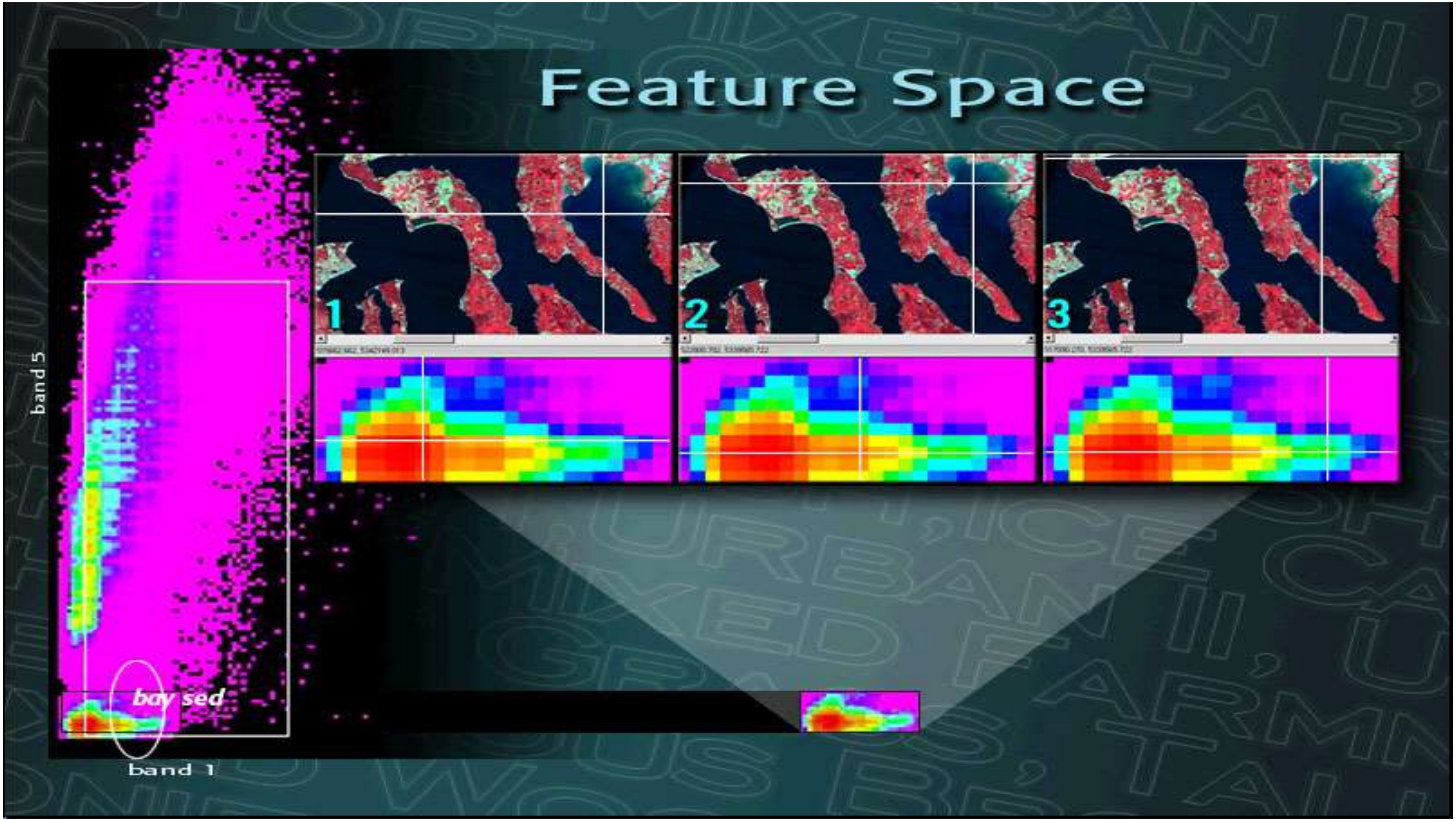
Hybrid Classification


superblocks
Forest & High Elevation
Urban, Suburban, Rural
Water, Shoreline, Wetland
Unsupervised clustering on combined superblocks dataset
Supervised training "pure" sites
Signature file
Spectral feature space plots & divergence
Editing & Merging Signature file
Classify Superblocks & others, print
Label, Describe based upon DOQ's, photos, ground
Combined & Labeled Signature file
Combine signatures statistics into one file
Final edit, Merge
Classification of path 46th
Class grouping (spatial label)
Map, Report

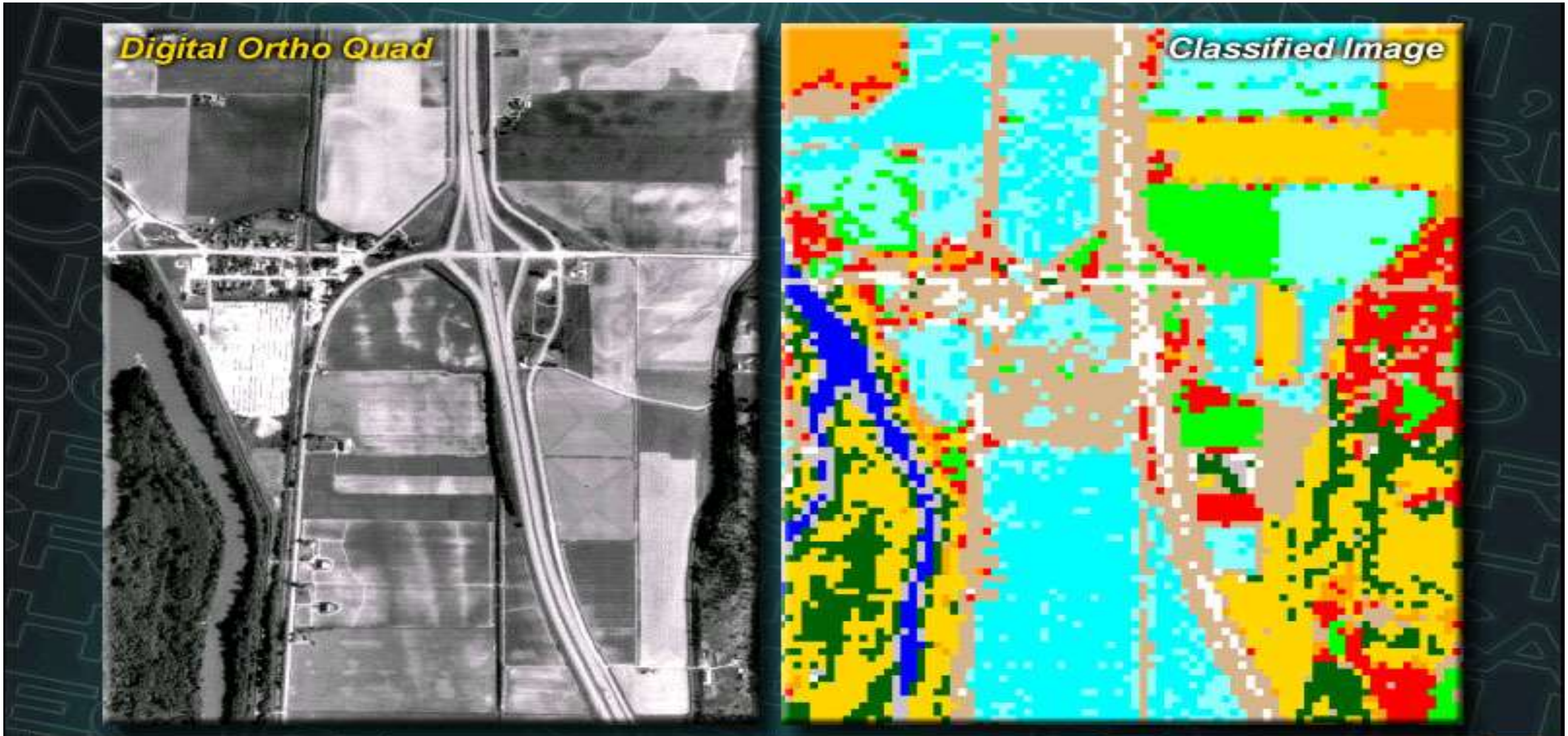
Hybrid - “superblocks”



Feature Space

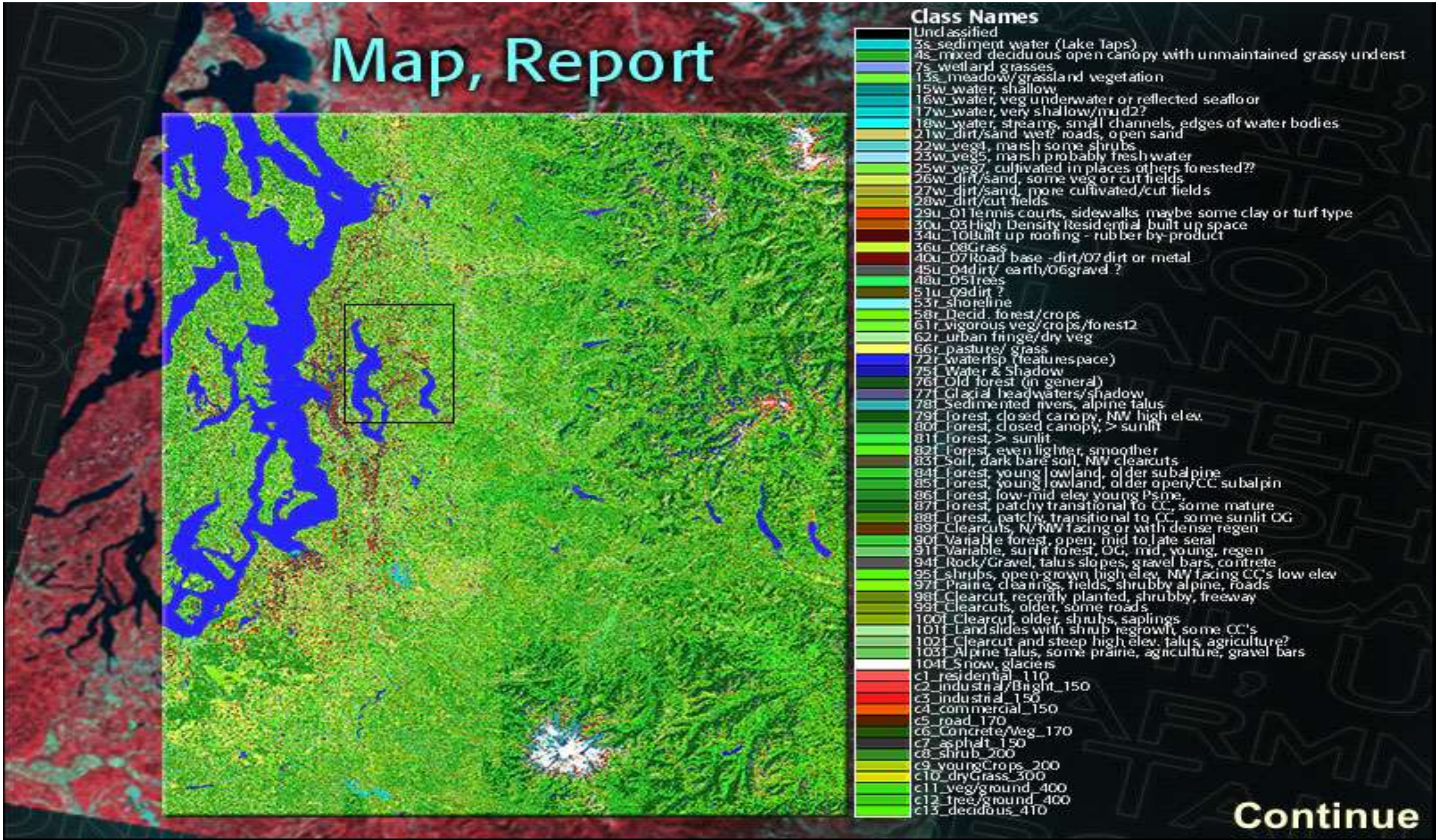


Ground Truth



Unclassified	⬛	shrub/forest	🟩	crops2 (trained)	🟧	bare/road	🟫
water	🟦	2nd growth/med.height trees	🟩	crops	🟧	young crops	🟥
waterfsp (featurespace)	🟦	dense forest	🟩	bare/drygrass	🟦	seedlings/young crops/bare	🟥
forestfsp (featurespace)	🟩	vigorous veg/crops/forest	🟨	dry/bare field (trained)	🟦	pasture/grass	🟩
forest (trained)	🟩	vigorous veg/crops/forest1	🟨	urban fringe/dry veg	🟫	shoreline	◽
Decid. forest/crops	🟩	vigorous veg/crops/forest2	🟨	bare/suburban/roadside/edge	🟫	urban/fringe shoreline	⬜

Classified Product



Class Names

Unclassified
3s_sediment water (Lake Taps)
4s_mixed deciduous open canopy with unmaintained grassy underst
7s_wetland grasses
13s_meadow/grassland vegetation
15w_water, shallow
16w_water, veg underwater or reflected seafloor
17w_water, very shallow/mud2?
18w_water, streams, small channels, edges of water bodies
21w_dirt/sand wet? roads, open sand
22w_veg4, marsh some shrubs
23w_veg5, marsh probably fresh water
25w_veg7, cultivated in places others forested??
26w_dirt/sand, some veg or cut fields
27w_dirt/sand, more cultivated/cut fields
28w_dirt/cut fields
29u_01Tennis courts, sidewalks maybe some clay or turf type
30u_03High Density Residential built up space
34u_10Built up roofing - rubber by-product
36u_08Grass
40u_07Road base -dirt/07dirt or metal
45u_04dirt/ earth/06gravel ?
48u_05Trees
51u_09dirt ?
53r_shoreline
58r_Decid. forest/crops
61r_vigorous veg/crops/forest2
62r_urban fringe/dry veg
66r_pasture/ grass
72r_waterfsp (featurespace)
75f_Water & Shadow
76f_Old forest (in general)
77f_Glacial headwaters/shadow
78f_Sedimented rivers, alpine talus
79f_forest, closed canopy, NW high elev.
80f_Forest, closed canopy, > sunlit
81f_forest, > sunlit
82f_Forest, even lighter, smoother
83f_Soil, dark bare soil, NW clearcuts
84f_Forest, young lowland, older subalpine
85f_Forest, young lowland, older open/CC subalpin
86f_Forest, low-mid elev young Psme,
87f_forest, patchy transitional to CC, some mature
88f_Forest, patchy, transitional to CC, some sunlit OG
89f_Clearcuts, N/NW facing or with dense regen
90f_Variable forest, open, mid to late seral
91f_Variable, sunlit forest, OG, mid, young, regen
94f_Rock/Gravel, talus slopes, gravel bars, contrete
95f_shrubs, open-grown high elev. NW facing CC's low elev
97f_Prairie, clearings, fields, shrubby alpine, roads
98f_Clearcut, recently planted, shrubby, freeway
99f_Clearcuts, older, some roads
100f_Clearcut, older, shrubs, saplings
101f_Landslides with shrub regrowth, some CC's
102f_Clearcut and steep high elev. talus, agriculture?
103f_Alpine talus, some prairie, agriculture, gravel bars
104f_Snow, glaciers
c1_residential_110
c2_industrial/Bright_150
c3_industrial_150
c4_commercial_150
c5_road_170
c6_Concrete/Veg_170
c7_asphalt_150
c8_shrub_200
c9_youngCrops_200
c10_dryGrass_300
c11_veg/ground_400
c12_tree/ground_400
c13_decidous_410

Unit-III
Nonparametric Methods

- Sign Test
- Wilcoxon Signed-Rank Test
- Mann-Whitney-Wilcoxon Test
- Kruskal-Wallis Test
- Rank Correlation

Nonparametric Methods

- Most of the statistical methods referred to as parametric require the use of interval- or ratio-scaled data.
- Nonparametric methods are often the only way to analyze nominal or ordinal data and draw statistical conclusions.
- Nonparametric methods require no assumptions about the population probability distributions.
- Nonparametric methods are often called distribution-free methods.

Nonparametric Methods

- In general, for a statistical method to be classified as nonparametric, it must satisfy at least one of the following conditions.
  - The method can be used with nominal data.
  - The method can be used with ordinal data.
  - The method can be used with interval or ratio data when no assumption can be made about the population probability distribution.

Sign Test

- A common application of the sign test involves using a sample of n potential customers to identify a preference for one of two brands of a product.
- The objective is to determine whether there is a difference in preference between the two items being compared.
- To record the preference data, we use a plus sign if the individual prefers one brand and a minus sign if the individual prefers the other brand.
- Because the data are recorded as plus and minus signs, this test is called the sign test.

Sign Test: Small-Sample Case

- The small-sample case for the sign test should be used whenever $n \leq 20$.
- The hypotheses are

$H_0: p = .50$ No preference for one brand over the other exists.

$H_a: p \neq .50$ A preference for one brand over the other exists.

- The number of plus signs is our test statistic.
- Assuming H_0 is true, the sampling distribution for the test statistic is a binomial distribution with $p = .5$.
- H_0 is rejected if the p-value $\leq$ level of significance, α.

Sign Test: Large-Sample Case

- Using H_0: p = .5 and n > 20, the sampling distribution for the number of plus signs can be approximated by a normal distribution.
- When no preference is stated (H_0: p = .5), the sampling distribution will have:

Mean: $\mu = .50n$

Standard Deviation: $\sigma = \sqrt{.25n}$

- The test statistic is: $$z = \frac{x - \mu}{\sigma}$$ (x is the number of plus signs)
- H_0 is rejected if the p-value ≤ level of significance, α.

Sign Test: Large-Sample Case

- Example: Ketchup Taste Test

As part of a market research study, a sample of 36 consumers were asked to taste two brands of ketchup and indicate a preference. Do the data shown on the next slide indicate a significant difference in the consumer preferences for the two brands?

Sign Test: Large-Sample Case

- Example: Ketchup Taste Test

18 preferred Brand A Ketchup
(+ sign recorded)

12 preferred Brand B Ketchup
(− sign recorded)

6 had no preference

The analysis will be based on
a sample size of 18 + 12 = 30.

Sign Test: Large-Sample Case

- Hypotheses

$H_0 : p = .50$

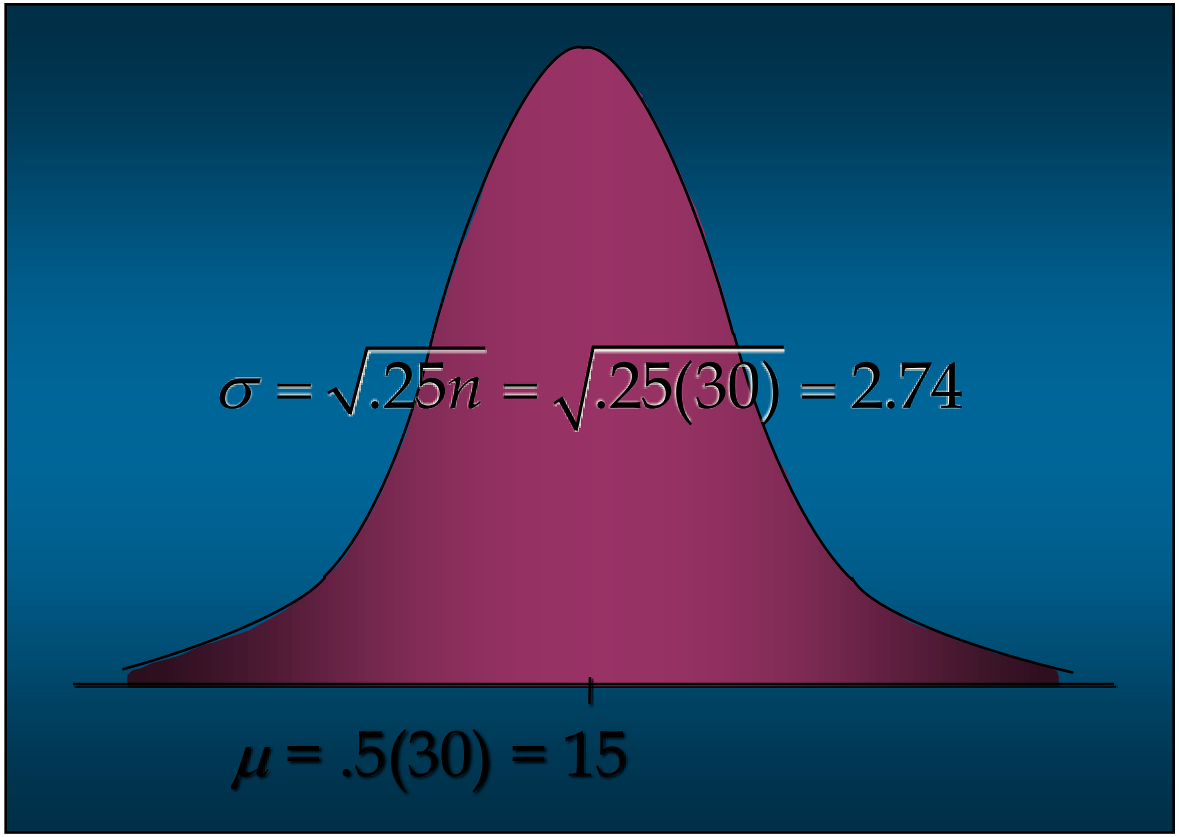
No preference for one brand over the other exists

$H_a : p \neq .50$

A preference for one brand over the other exists

Sign Test: Large-Sample Case

- Sampling Distribution for Number of Plus Sig



$$\sigma = \sqrt{.25n} = \sqrt{.25(30)} = 2.74$$

$$\mu = .5(30) = 15$$

Sign Test: Large-Sample Case

- Rejection Rule

Using .05 level of significance:

Reject H_0 if p-value $\leq$ .05

- Test Statistic

$$z = (x - \mu)/\sigma = (18 - 15)/2.74 = 3/2.74 = \boxed{1.10}$$

- p-Value

$$p\text{-Value} = 2(.5000 - .3643) = \boxed{.2714}$$

Sign Test: Large-Sample Case

- Conclusion

Because the *p*-value > α, we cannot reject H_0. There is insufficient evidence in the sample to conclude that a difference in preference exists for the two brands of ketchup.

Hypothesis Test About a Median

- We can apply the sign test by:
  - Using a plus sign whenever the data in the sample are above the hypothesized value of the median
  - Using a minus sign whenever the data in the sample are below the hypothesized value of the median
  - Discarding any data exactly equal to the hypothesized median

Hypothesis Test About a Median

- Example: Trim Fitness Center

- A hypothesis test is being conducted about the median age of female members of the Trim Fitness Center.

- H_0: Median Age = 34 years
  H_a: Median Age ≠ 34 years

- In a sample of 40 female members, 25 are older than 34, 14 are younger than 34, and 1 is 34. Is there sufficient evidence to reject H_0? Assume α = .05.

Hypothesis Test About a Median

- Mean and Standard Deviation

$$\mu = .5(39) = 19.5$$

$$\sigma = \sqrt{.25n} = \sqrt{.25(39)} = 3.12$$

- Test Statistic

$$z = (x - \mu)/\sigma = (25 - 19.5)/3.12 = \boxed{1.76}$$

- *p*-Value

$$p\text{-Value} = 2(.5000 - .4608) = \boxed{.0784}$$

Hypothesis Test About a Median

- Rejection Rule

Using .05 level of significance:

Reject H_0 if p-value $\leq$ .05

- Conclusion

Do not reject H_0. The p-value for this two-tail test is .0784. There is insufficient evidence in the sample to conclude that the median age is not 34 for female members of Trim Fitness Center.

Wilcoxon Signed-Rank Test

- This test is the nonparametric alternative to the parametric matched-sample test presented in Chapter 10.
- The methodology of the parametric matched-sample analysis requires:
  - interval data, and
  - the assumption that the population of differences between the pairs of observations is normally distributed.
- If the assumption of normally distributed differences is not appropriate, the Wilcoxon signed-rank test can be used.

Wilcoxon Signed-Rank Test

- Example: Express Deliveries

- A firm has decided to select one of two express delivery services to provide next-day deliveries to its district offices.

  To test the delivery times of the two services, the firm sends two reports to a sample of 10 district offices, with one report carried by one service and the other report carried by the second service. Do the data on the next slide indicate a difference in the two services?

Wilcoxon Signed-Rank Test

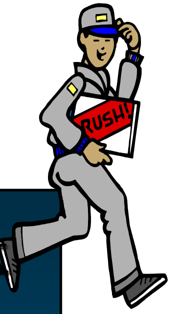
District Office	OverNight	NiteFlite
Seattle	32 hrs.	25 hrs.
Los Angeles	30	24
Boston	19	15
Cleveland	16	15
New York	15	13
Houston	18	15
Atlanta	14	15
St. Louis	10	8
Milwaukee	7	9
Denver	16	11

Wilcoxon Signed-Rank Test

- Preliminary Steps of the Test
  - Compute the differences between the paired observations.
  - Discard any differences of zero.
  - Rank the absolute value of the differences from lowest to highest. Tied differences are assigned the average ranking of their positions.
  - Give the ranks the sign of the original difference in the data.
  - Sum the signed ranks.

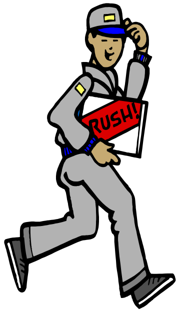
. . . next we will determine whether the sum is significantly different from zero.

Wilcoxon Signed-Rank Test



District Office	Differ.	|Diff.| Rank	Sign. Rank
Seattle	7	10	+10
Los Angeles	6	9	+9
Boston	4	7	+7
Cleveland	1	1.5	+1.5
New York	2	4	+4
Houston	3	6	+6
Atlanta	–1	1.5	–1.5
St. Louis	2	4	+4
Milwaukee	–2	4	–4
Denver	5	8	+8
			+44

Wilcoxon Signed-Rank Test

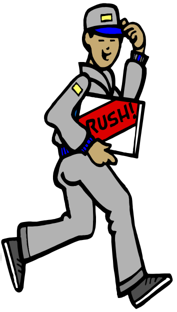


- Hypotheses

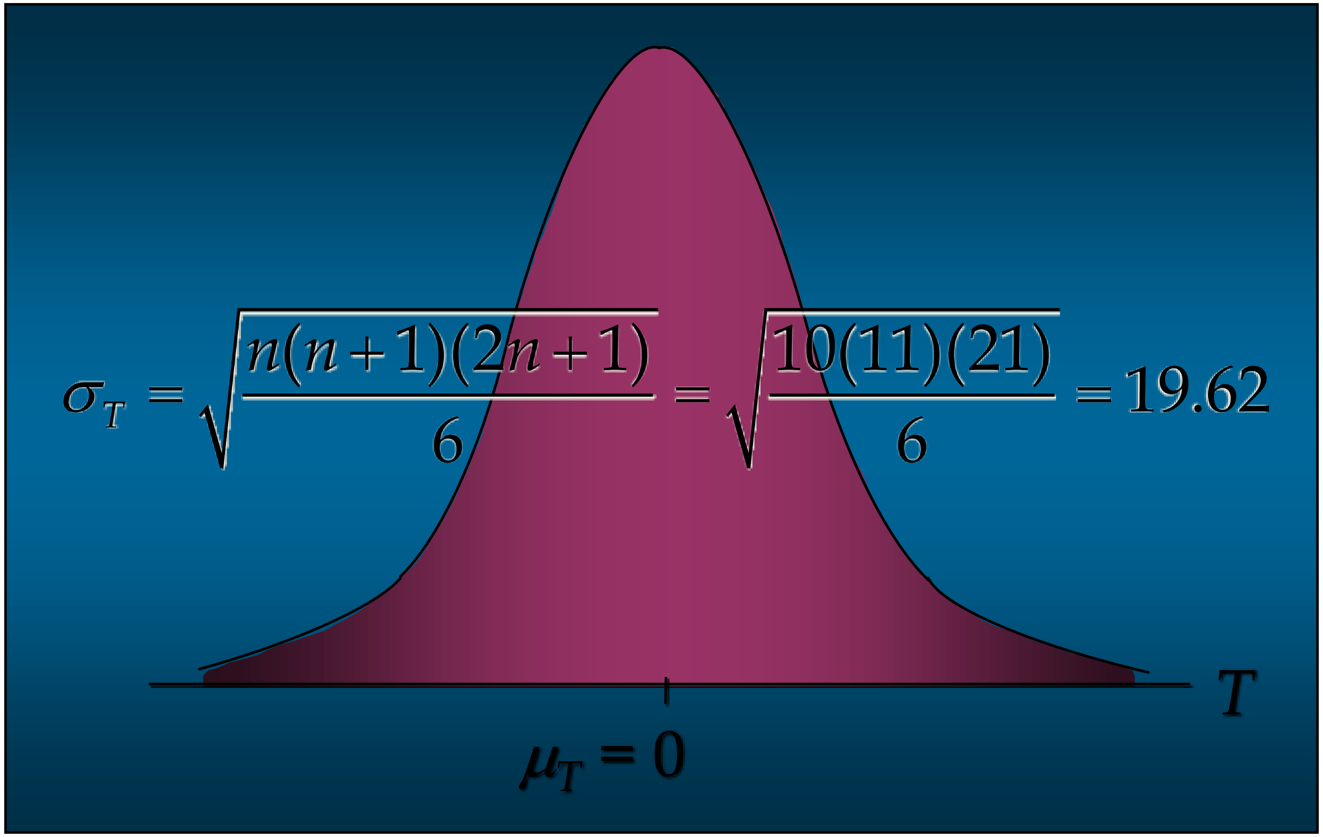
  - H_0: The delivery times of the two services are the same; neither offers faster service than the other.

    H_a: Delivery times differ between the two services; recommend the one with the smaller times.

Wilcoxon Signed-Rank Test

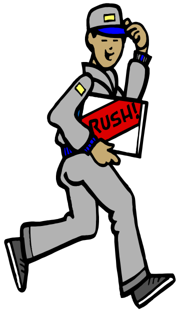


- Sampling Distribution of *T* for Identical Populations



$$\sigma_T = \sqrt{\frac{n(n+1)(2n+1)}{6}} = \sqrt{\frac{10(11)(21)}{6}} = 19.62$$

Wilcoxon Signed-Rank Test



- Rejection Rule

Using .05 level of significance,

Reject H_0 if p-value $\leq .05$

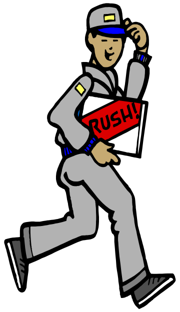
- Test Statistic

$$z = (T - \mu_T)/\sigma_T = (44 - 0)/19.62 = 2.24$$

- p-Value

$$p\text{-Value} = 2(.5000 - .4875) = .025$$

Wilcoxon Signed-Rank Test



- Conclusion

  Reject H_0. The p-value for this two-tail test is .025. There is sufficient evidence in the sample to conclude that a difference exists in the delivery times provided by the two services.

Mann-Whitney-Wilcoxon Test

- This test is another nonparametric method for determining whether there is a difference between two populations.
- This test, unlike the Wilcoxon signed-rank test, is not based on a matched sample.
- This test does not require interval data or the assumption that both populations are normally distributed.
- The only requirement is that the measurement scale for the data is at least ordinal.

Mann-Whitney-Wilcoxon Test

- Instead of testing for the difference between the means of two populations, this method tests to determine whether the two populations are identical.

- The hypotheses are:

H_0: The two populations are identical

H_a: The two populations are not identical

Mann-Whitney-Wilcoxon Test

- Example: Westin Freezers

Manufacturer labels indicate the annual energy cost associated with operating home appliances such as freezers.

The energy costs for a sample of 10 Westin freezers and a sample of 10 Easton Freezers are shown on the next slide. Do the data indicate, using $\alpha = .05$, that a difference exists in the annual energy costs for the two brands of freezers?

Mann-Whitney-Wilcoxon Test

Westin Freezers	Easton Freezers
$55.10	$56.10
54.50	54.70
53.20	54.40
53.00	55.40
55.50	54.10
54.90	56.00
55.80	55.50
54.00	55.00
54.20	54.30
55.20	57.00

Mann-Whitney-Wilcoxon Test

- Hypotheses

H_0: Annual energy costs for Westin freezers and Easton freezers are the same.

H_a: Annual energy costs differ for the two brands of freezers.

Mann-Whitney-Wilcoxon Test: Large-Sample Case

- First, rank the combined data from the lowest to the highest values, with tied values being assigned the average of the tied rankings.
- Then, compute *T*, the sum of the ranks for the first sample.
- Then, compare the observed value of *T* to the sampling distribution of *T* for identical populations. The value of the standardized test statistic *z* will provide the basis for deciding whether to reject H_0.

Mann-Whitney-Wilcoxon Test: Large-Sample Case

- Sampling Distribution of *T* for Identical Populations
  - ▶ • Mean

$\mu_T =$ 🗀 $n_1(n_1 + n_2 + 1)$

  - ▶ • Standard Deviation

$$\sigma_T = \sqrt{\tfrac{1}{12} n_1 n_2 (n_1 + n_2 + 1)}$$

  - ▶ • Distribution Form

Approximately normal, provided

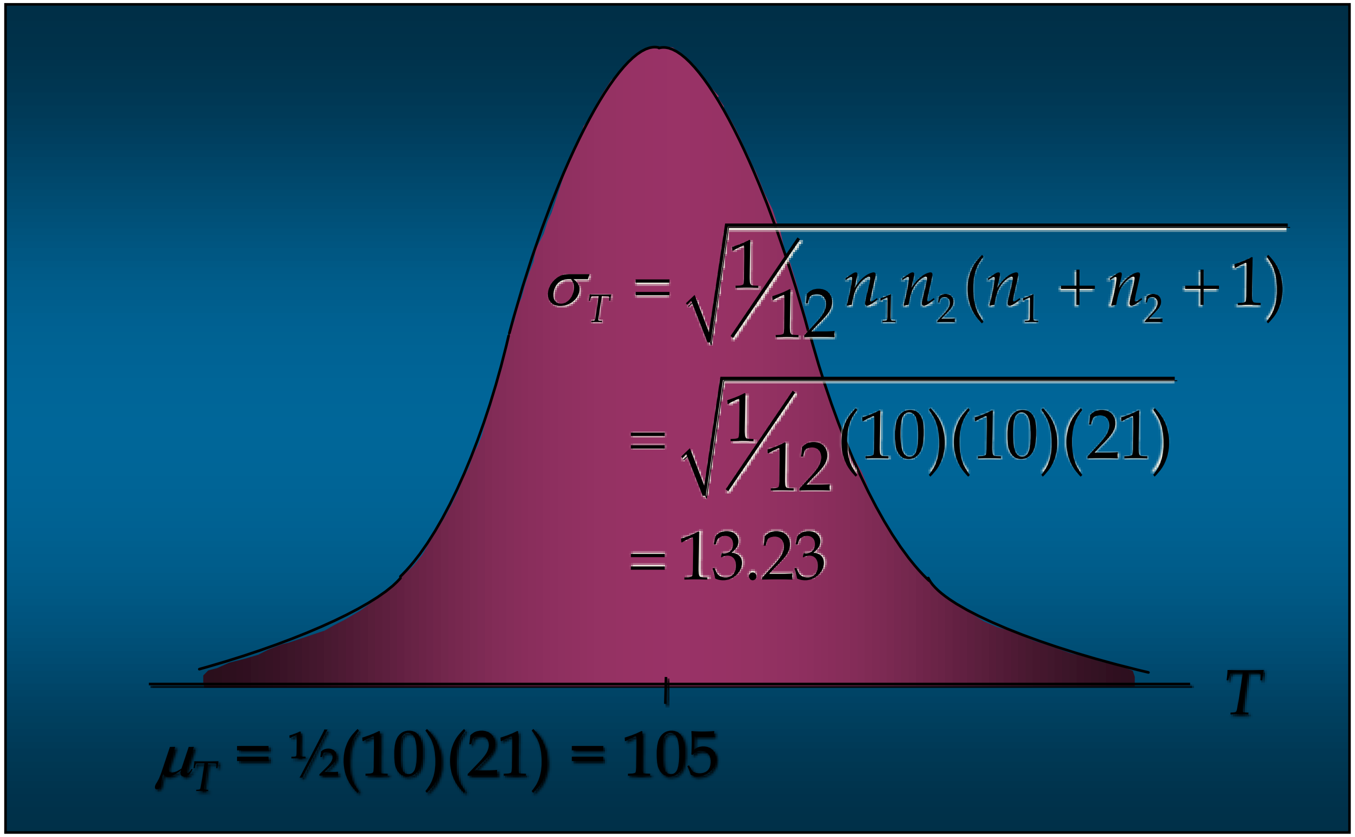
$n_1 \geq 10$ **and** $n_2 \geq 10$

Mann-Whitney-Wilcoxon Test

Westin Freezers	Rank	Easton Freezers	Rank
$55.10	12	$56.10	19
54.50	8	54.70	9
53.20	2	54.40	7
53.00	1	55.40	14
55.50	15.5	54.10	4
54.90	10	56.00	18
55.80	17	55.50	15.5
54.00	3	55.00	11
54.20	5	54.30	6
55.20	13	57.00	20
Sum of Ranks	86.5	Sum of Ranks	123.5

Mann-Whitney-Wilcoxon Test

- Sampling Distribution of *T* for Identical Populations



$$\sigma_T = \sqrt{\tfrac{1}{12} n_1 n_2 (n_1 + n_2 + 1)}$$

$$= \sqrt{\tfrac{1}{12}(10)(10)(21)}$$

$$= 13.23$$

μ_T = ½(10)(21) = 105

Mann-Whitney-Wilcoxon Test

- Rejection Rule

Using .05 level of significance,

Reject H_0 if p-value $\leq .05$

- Test Statistic

$$z = (T - \mu_T)/\sigma_T = (86.5 - 105)/13.23 = \boxed{-1.40}$$

- *p*-Value

$$p\text{-Value} = 2(.5000 - .4192) = \boxed{.1616}$$

Mann-Whitney-Wilcoxon Test

- Conclusion

Do not reject H_0. The p-value > α. There is insufficient evidence in the sample data to conclude that there is a difference in the annual energy cost associated with the two brands of freezers.

Kruskal-Wallis Test

- The Mann-Whitney-Wilcoxon test has been extended by Kruskal and Wallis for cases of three or more populations.

H_0: All populations are identical

H_a: Not all populations are identical

- The Kruskal-Wallis test can be used with ordinal data as well as with interval or ratio data.
- Also, the Kruskal-Wallis test does not require the assumption of normally distributed populations.

Kruskal-Wallis Test

- Test Statistic

$$W = \left[\frac{12}{n_T(n_T+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} \right] - 3(n_T+1)$$

where: k = number of populations

n_i = number of items in sample i

$n_T = \Sigma n_i$ = total number of items in all samples

R_i = sum of the ranks for sample i

Kruskal-Wallis Test

- When the populations are identical, the sampling distribution of the test statistic W can be approximated by a chi-square distribution with $k - 1$ degrees of freedom.

- This approximation is acceptable if each of the sample sizes n_i is $\geq$ 5.

- The rejection rule is: Reject H_0 if p-value $\leq \alpha$

Rank Correlation

- The Pearson correlation coefficient, *r*, is a measure of the linear association between two variables for which interval or ratio data are available.
- The Spearman rank-correlation coefficient, r_s , is a measure of association between two variables when only ordinal data are available.
- Values of r_s can range from –1.0 to +1.0, where
  - values near 1.0 indicate a strong positive association between the rankings, and
  - values near -1.0 indicate a strong negative association between the rankings.

Rank Correlation

- Spearman Rank-Correlation Coefficient, r_s

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

where: n = number of items being ranked

x_i = rank of item i with respect to one variable

y_i = rank of item i with respect to a second variable

$d_i = x_i - y_i$

Test for Significant Rank Correlation

- We may want to use sample results to make an inference about the population rank correlation p_s.
- To do so, we must test the hypotheses:

$$H_0 : p_s = 0 \quad \text{(No rank correlation exists)}$$
$$H_a : p_s \neq 0 \quad \text{(Rank correlation exists)}$$

Rank Correlation

- Sampling Distribution of r_s when $p_s = 0$
  - Mean

$$\boxed{\mu_{r_s} = 0}$$

  - Standard Deviation

$$\boxed{\sigma_{r_s} = \sqrt{\frac{1}{n-1}}}$$

  - Distribution Form

Approximately normal, provided $n \geq 10$

Rank Correlation

- Example: Crennor Investors

Crennor Investors provides a portfolio management service for its clients. Two of Crennor's analysts ranked ten investments as shown on the next slide. Use rank correlation, with $\alpha = .10$, to comment on the agreement of the two analysts' rankings.

Rank Correlation

- Example: Crennor Investors
  - Analysts' Rankings

Investment	A	B	C	D	E	F	G	H	I	J
Analyst #1	1	4	9	8	6	3	5	7	2	10
Analyst #2	1	5	6	2	9	7	3	10	4	8

  - Hypotheses

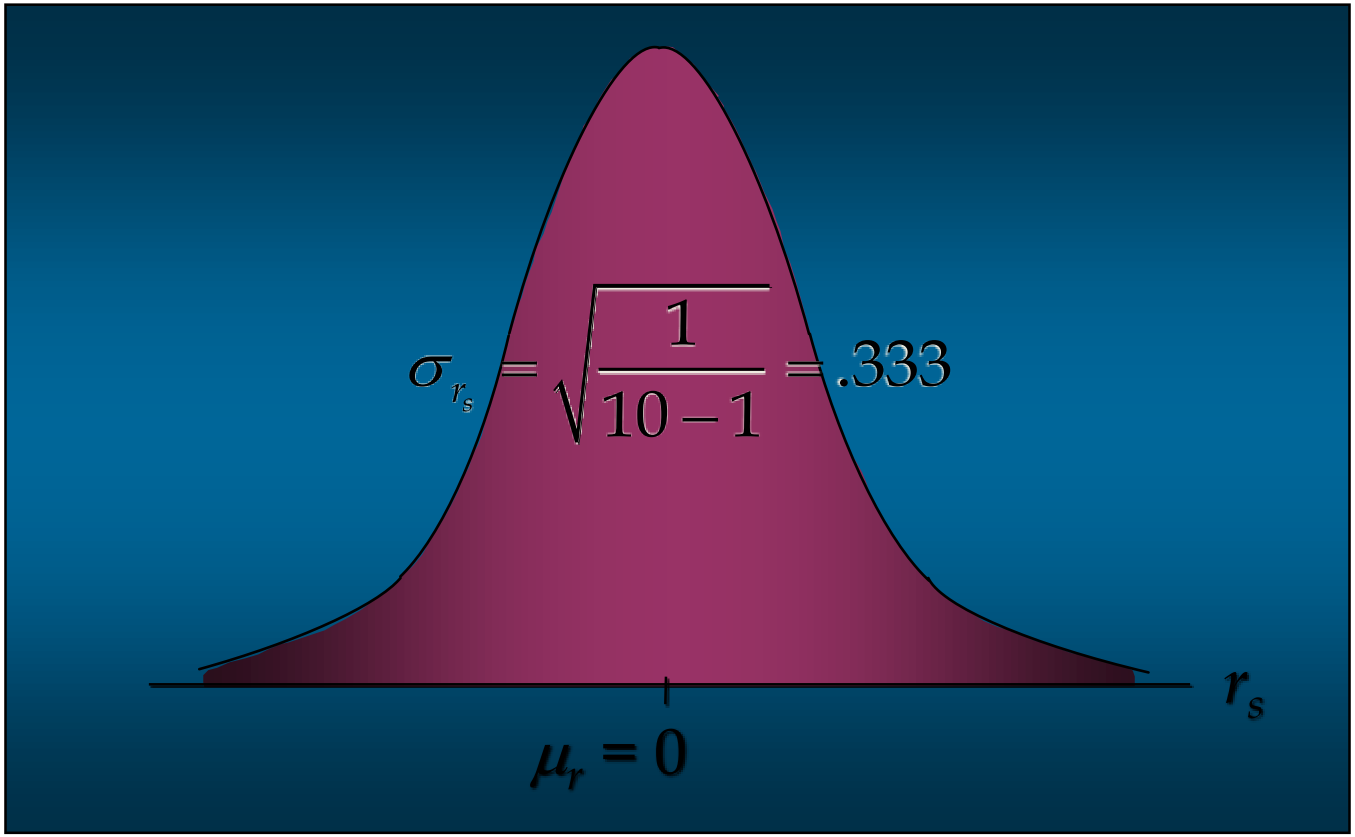
$$H_0 : p_s = 0 \quad \text{(No rank correlation exists)}$$
$$H_a : p_s \neq 0 \quad \text{(Rank correlation exists)}$$

Rank Correlation

	Analyst #1	Analyst #2		
Investment	Ranking	Ranking	Differ.	(Differ.)^2
A	1	1	0	0
B	4	5	-1	1
C	9	6	3	9
D	8	2	6	36
E	6	9	-3	9
F	3	7	-4	16
G	5	3	2	4
H	7	10	-3	9
I	2	4	-2	4
J	10	8	2	4
				Sum = 92

Rank Correlation

- Sampling Distribution of r_s Assuming No Rank Correlation



$$\sigma_{r_s} = \sqrt{\frac{1}{10-1}} = .333$$

Rank Correlation

- Rejection Rule

With .10 level of significance:

Reject H_0 if p-value $\leq$ .10

- Test Statistic

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)} = 1 - \frac{6(92)}{10(100-1)} = 0.4424$$

$$z = (r_s - \mu_r)/\sigma_r = (.4424 - 0)/.3333 = 1.33$$

- *p*-Value

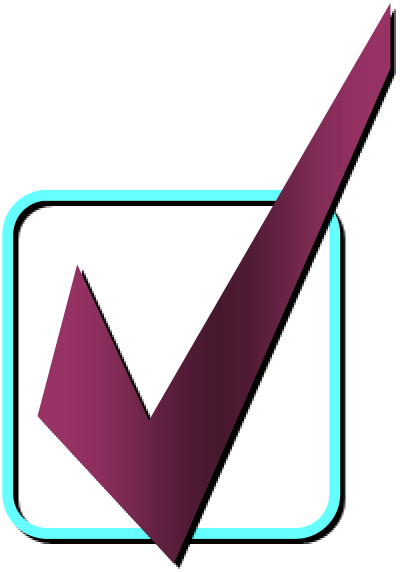
p-Value = 2(.5000 - .4082) = .1836

Rank Correlation

- Conclusion

- Do no reject H_0. The p-value > α. There is not a significant rank correlation. The two analysts are not showing agreement in their ranking of the risk associated with the different investments.

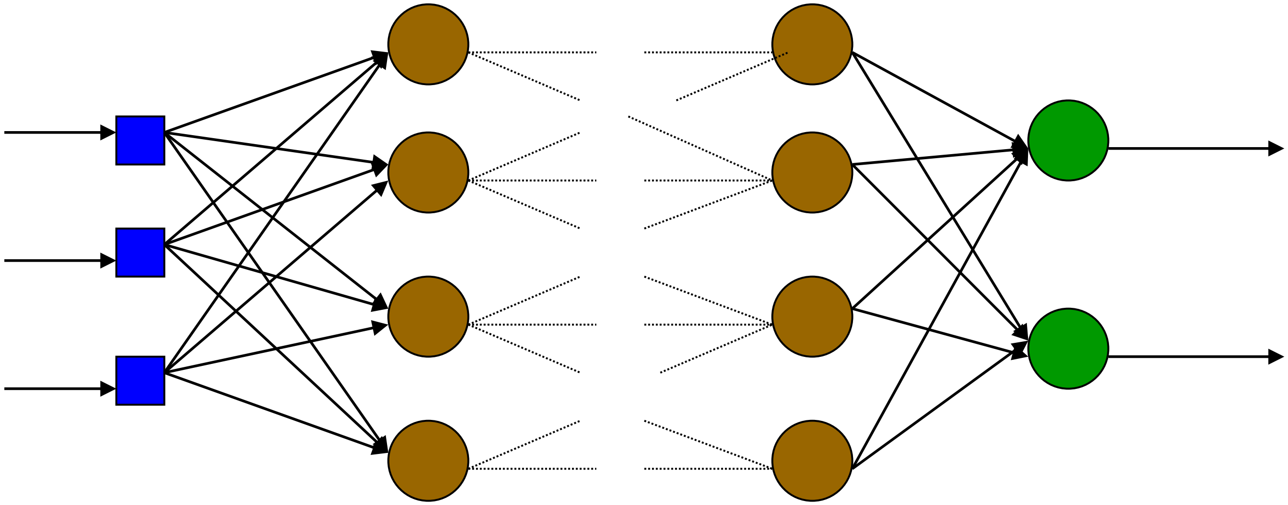
End of Chapter



UNIT- IV

Multilayer Percetrons

Multilayer Perceptrons Architecture



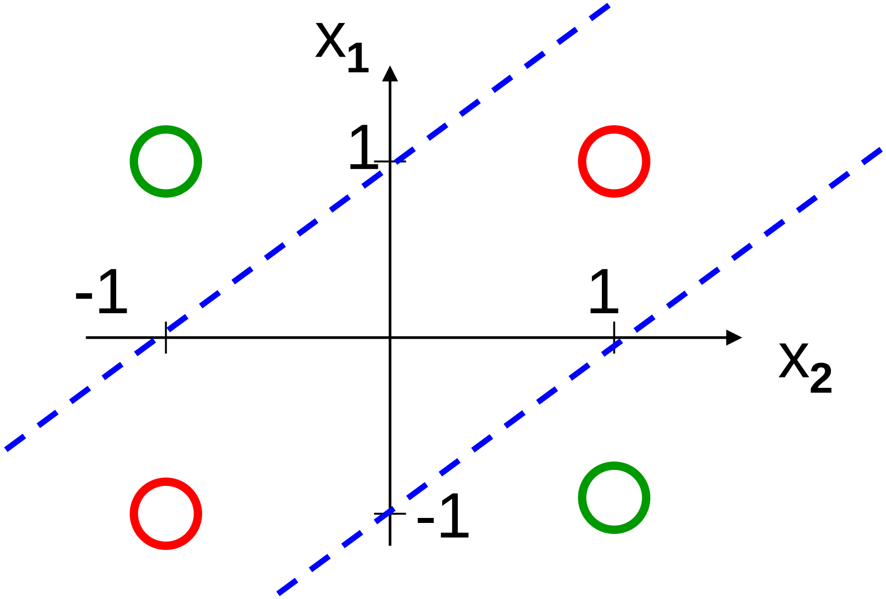
Input layer

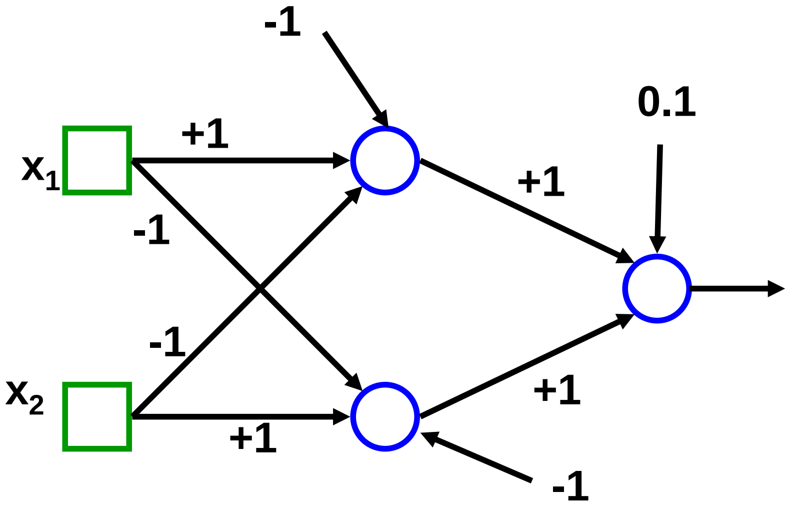
Output layer

Hidden Layers

A solution for the XOR problem

x_1	x_2	x_1 xor x_2
-1	-1	**-1**
-1	1	**1**
1	-1	**1**
1	1	**-1**



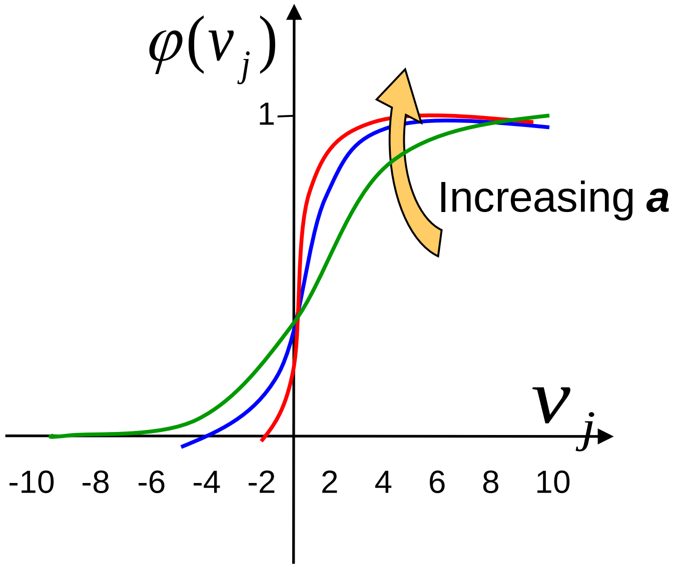


$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ -1 & \text{if } v \leq 0 \end{cases}$$

φ is the sign function.

NEURON MODEL

- Sigmoidal Function



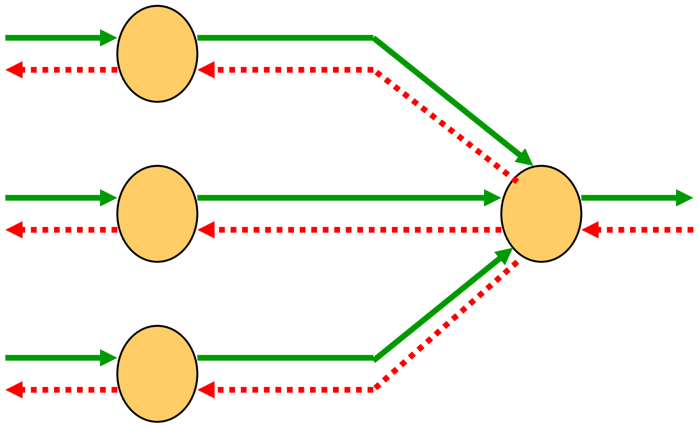
$$\boxed{\varphi(\mathrm{v}_j) = \frac{1}{1+e^{-av_j}}}$$

$$\mathrm{v}_j = \sum_{i=0,\ldots,m} w_{ji} y_i$$

- v_j induced field of neuron j
- Most common form of activation function
- a →∞ ⇒ φ → threshold function
- Differentiable

LEARNING ALGORITHM

- Back-propagation algorithm



→ *Function signals*
Forward Step

⇠ *Error signals*
Backward Step

- It adjusts the weights of the NN in order to minimize the average squared error.

Average Squared Error

- Error signal of output neuron ***j*** at presentation of ***n-th*** training example:

$$\boxed{e_j(n) = d_j(n) - y_j(n)}$$

- Total energy at time ***n***:

$$\boxed{E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n)}$$

- Average squared error:

- Measure of learning performance:

$$\boxed{E_{AV} = \frac{1}{N}\sum_{n=1}^{N} E(n)}$$

C: Set of neurons in output layer

N: size of training set

- **Goal:** ***Adjust weights of NN to minimize E_{AV}***

Notation

e_{j} Error at output of neuron j

y_{j} Output of neuron j

$$\mathrm{v}_{\mathrm{j}} = \sum_{i=0,\ldots,m} w_{\mathrm{ji}} y_{\mathrm{i}}$$

Induced local field of neuron j

Weight Update Rule

Update rule is based on the gradient descent method take a step in the direction yielding the maximum decrease of E

$$\boxed{\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}}$$

Step in direction opposite to the gradient

With w_{ji} weight associated to the link from neuron i to neuron j

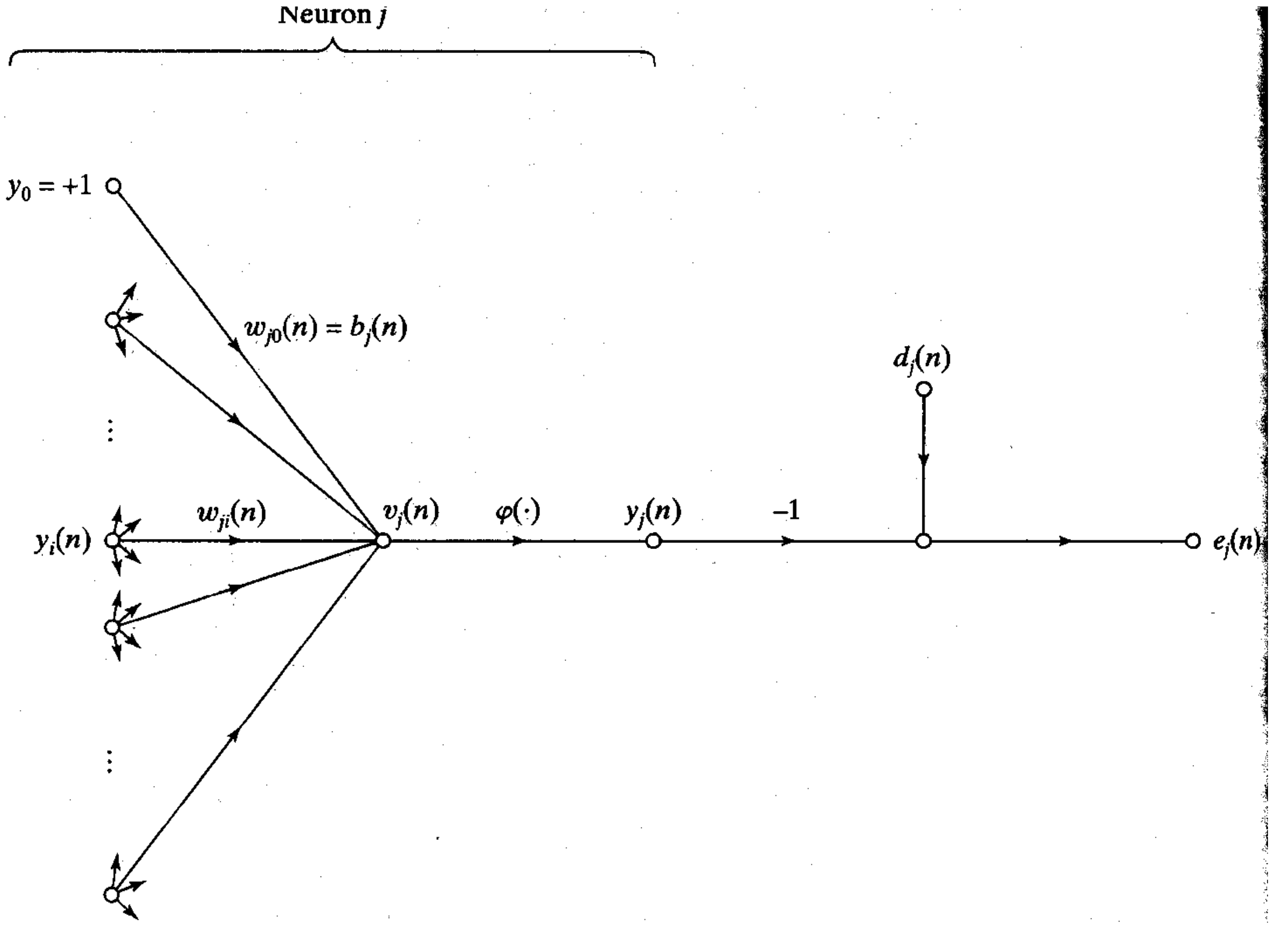


FIGURE 4.3 Signal-flow graph highlighting the details of output neuron *j*.

Definition of the Local Gradient of neuron j

$$\boxed{\delta_j = -\frac{\partial E}{\partial v_j}}$$

Local Gradient

We obtain $\boxed{\delta_j = e_j \varphi'(v_j)}$

because

$$-\frac{\partial E}{\partial v_j} = -\frac{\partial E}{\partial e_j}\frac{\partial e_j}{\partial y_j}\frac{\partial y_j}{\partial v_j} = -e_j(-1)\varphi'(v_j)$$

Update Rule

- We obtain

$$\boxed{\Delta \mathrm{w}_{\mathrm{ji}} = \eta \delta_j y_i}$$

because

$$\frac{\partial E}{\partial \mathrm{w}_{\mathrm{ji}}} = \frac{\partial E}{\partial \mathrm{v}_{\mathrm{j}}} \frac{\partial \mathrm{v}_{\mathrm{j}}}{\partial \mathrm{w}_{\mathrm{ji}}}$$

$$-\frac{\partial E}{\partial \mathrm{v}_{j}} = \delta_j \qquad \frac{\partial \mathrm{v}_{j}}{\partial \mathrm{w}_{\mathrm{ji}}} = y_i$$

Compute local gradient of neuron j

- The key factor is the calculation of $\boldsymbol{e_j}$
- There are two cases:
  - Case 1): $\boldsymbol{j}$ is a output neuron
  - Case 2): $\boldsymbol{j}$ is a hidden neuron

Error e_j of output neuron

- Case 1: ***j output neuron***

$$e_j = d_j - y_j$$

Then

$$\boxed{\delta_j = (d_j - y_j)\varphi'(v_j)}$$

Local gradient of hidden neuron

- Case 2: ***j hidden neuron***

- the local gradient for neuron j is recursively determined in terms of the local gradients of all neurons to which neuron j is directly connected

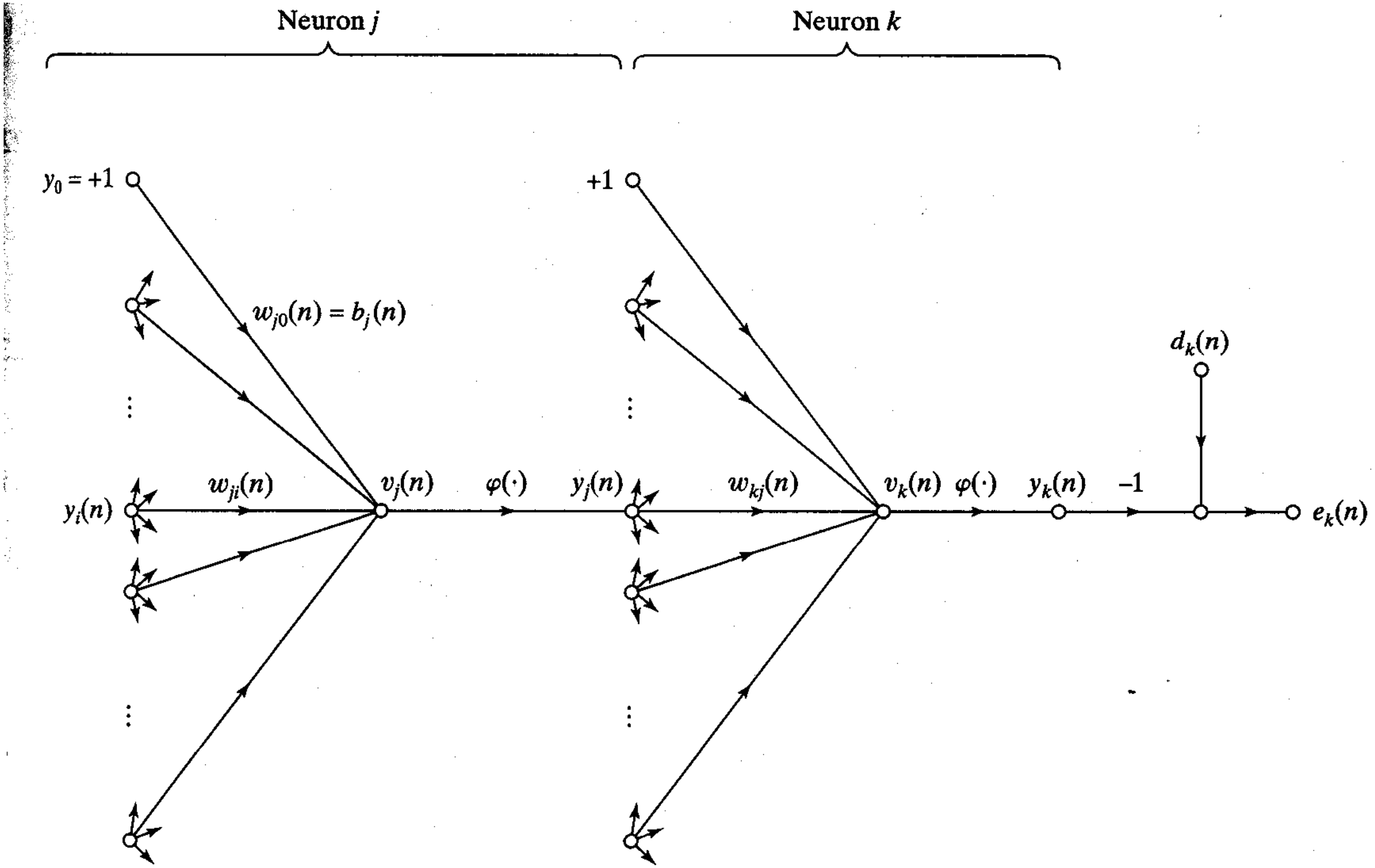


FIGURE 4.4 Signal-flow graph highlighting the details of output neuron k connected to hidden neuron j.

Use the Chain Rule

$$\delta_j = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_j} \qquad\qquad \frac{\partial y_j}{\partial v_j} = \varphi'(v_j)$$

$$\boxed{E(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n)}$$

$$-\frac{\partial E}{\partial y_j} = -\sum_{k\in C} e_k \frac{\partial e_k}{\partial y_j} = \sum_{k\in C} e_k \left[\frac{-\partial e_k}{\partial v_k} \right] \frac{\partial v_k}{\partial y_j}$$

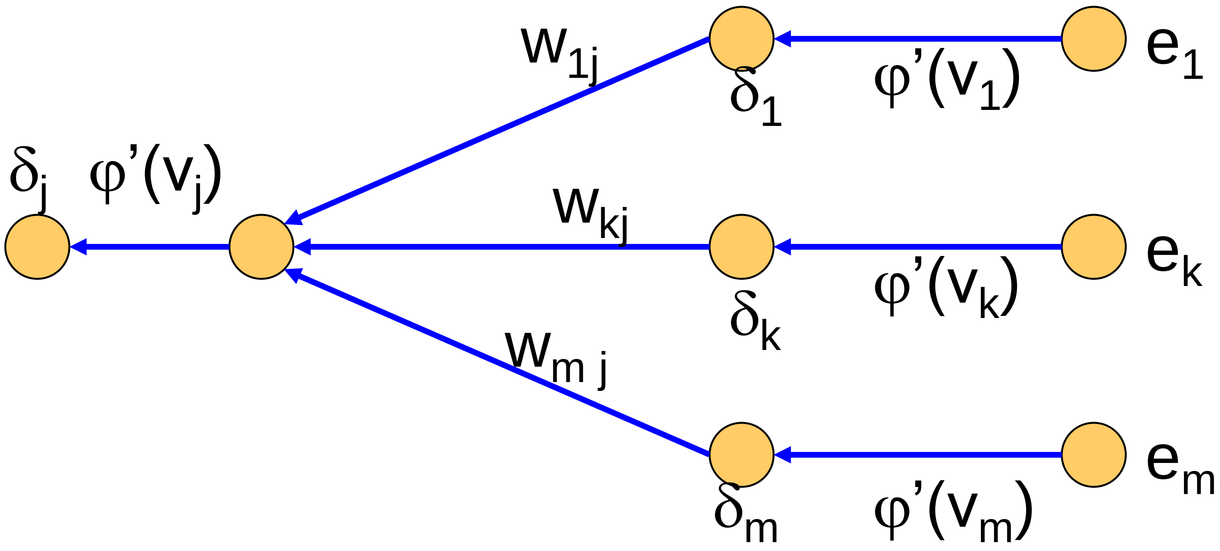
from $\quad -\frac{\partial e_k}{\partial v_k} = \varphi'(v_k) \qquad \frac{\partial v_k}{\partial y_j} = w_{kj}$

We obtain $\quad -\frac{\partial E}{\partial y_j} = \sum_{k\in C} \delta_k w_{kj}$

Local Gradient of hidden neuron j

Hence

$$\boxed{\delta_j = \varphi'(v_j)\sum_{k\in C}\delta_k w_{kj}}$$



Signal-flow graph of back-propagation error signals to neuron ***j***

Delta Rule

- **Delta rule** $\Delta w_{ji} = \eta \delta_j y_i$

$$\delta_j = \begin{cases} \varphi'(v_j)(d_j - y_j) & \text{IF j output node} \\ \varphi'(v_j)\sum_{k \in C} \delta_k w_{kj} & \text{IF j hidden node} \end{cases}$$

C: Set of neurons in the layer following the one containing ***j***

Local Gradient of neurons

$$\varphi'(v_j) = a y_j [1 - y_j]$$

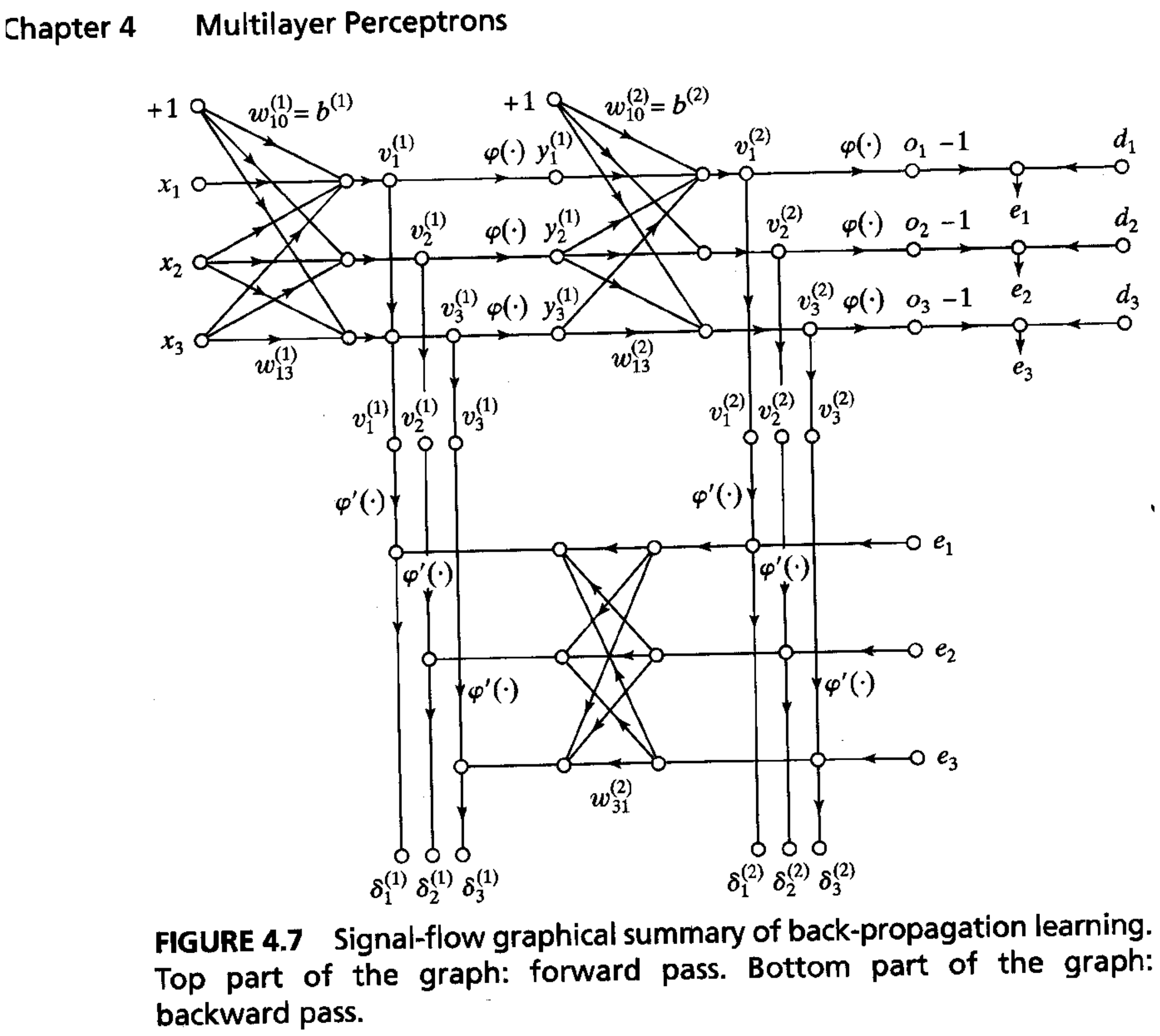
a > 0

$$\delta_j = \begin{cases} a y_j [1 - y_j] \sum \delta_k w_{kj} & \textit{if j hidden node} \\ a y_j [1 - y_j][d_j^k - y_j] & \textit{If j output node} \end{cases}$$

Backpropagation algorithm

- Two phases of computation:
  - **Forward pass**: run the NN and compute the error for each neuron of the output layer.
  - **Backward pass**: start at the output layer, and pass the errors backwards through the network, layer by layer, by recursively computing the local gradient of each neuron.

Summary



Chapter 4 Multilayer Perceptrons

FIGURE 4.7 Signal-flow graphical summary of back-propagation learning. Top part of the graph: forward pass. Bottom part of the graph: backward pass.

Training

- **Sequential mode** (on-line, pattern or stochastic mode):
  - (x(1), d(1)) is presented, a sequence of forward and backward computations is performed, and the weights are updated using the delta rule.
  - Same for (x(2), d(2)), … , (x(N), d(N)).

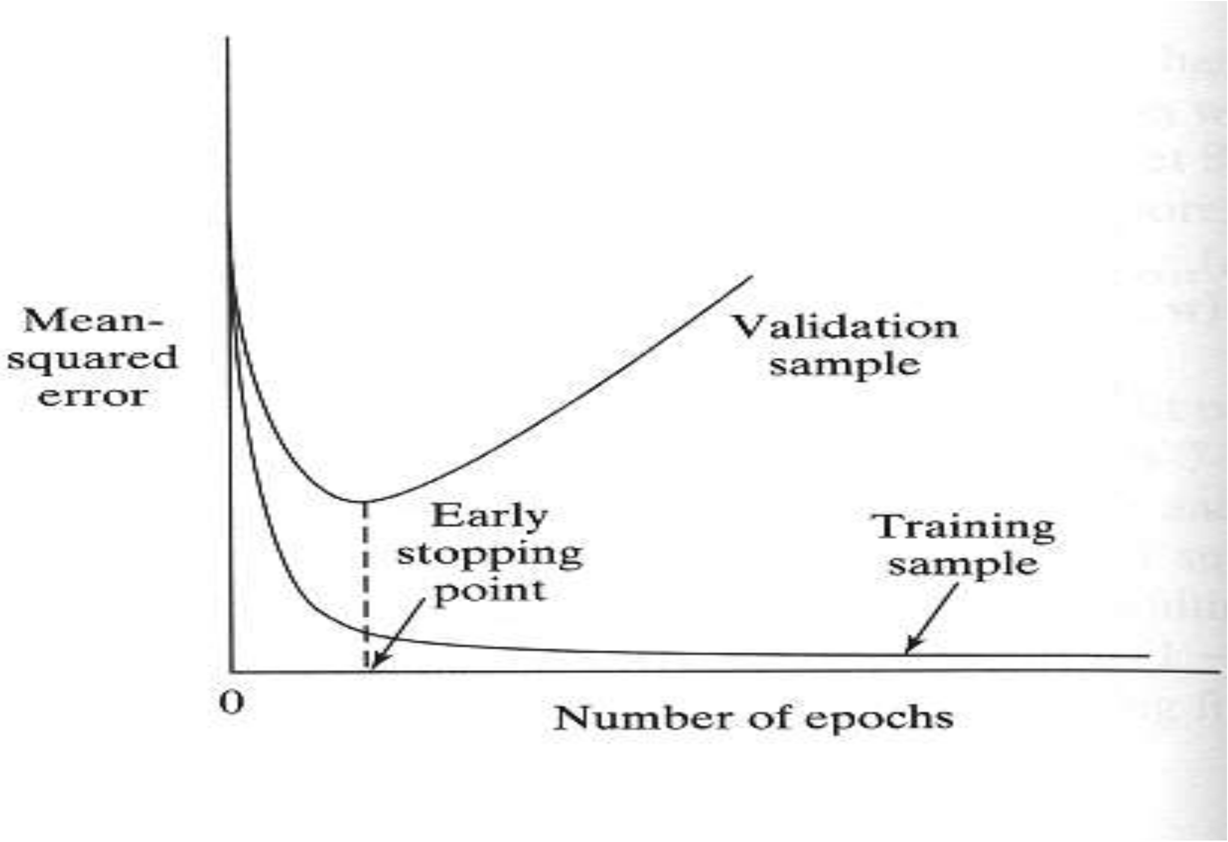
Training

- The learning process continues on an epoch-by-epoch basis until the stopping condition is satisfied.
- From one epoch to the next choose a randomized ordering for selecting examples in the training set.

Stopping criterions

- Sensible stopping criterions:
  - Average squared error change: Back-prop is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small (in the range [0.1, 0.01]).
  - Generalization based criterion: After each epoch the NN is tested for generalization. If the generalization performance is adequate then stop.

Early stopping



Generalization

- Generalization: NN generalizes well if the I/O mapping computed by the network is nearly correct for new data (test set).
- Factors that influence generalization:
  - the size of the training set.
  - the architecture of the NN.
  - the complexity of the problem at hand.
- Overfitting (overtraining): when the NN learns too many I/O examples it may end up memorizing the training data.

Generalization

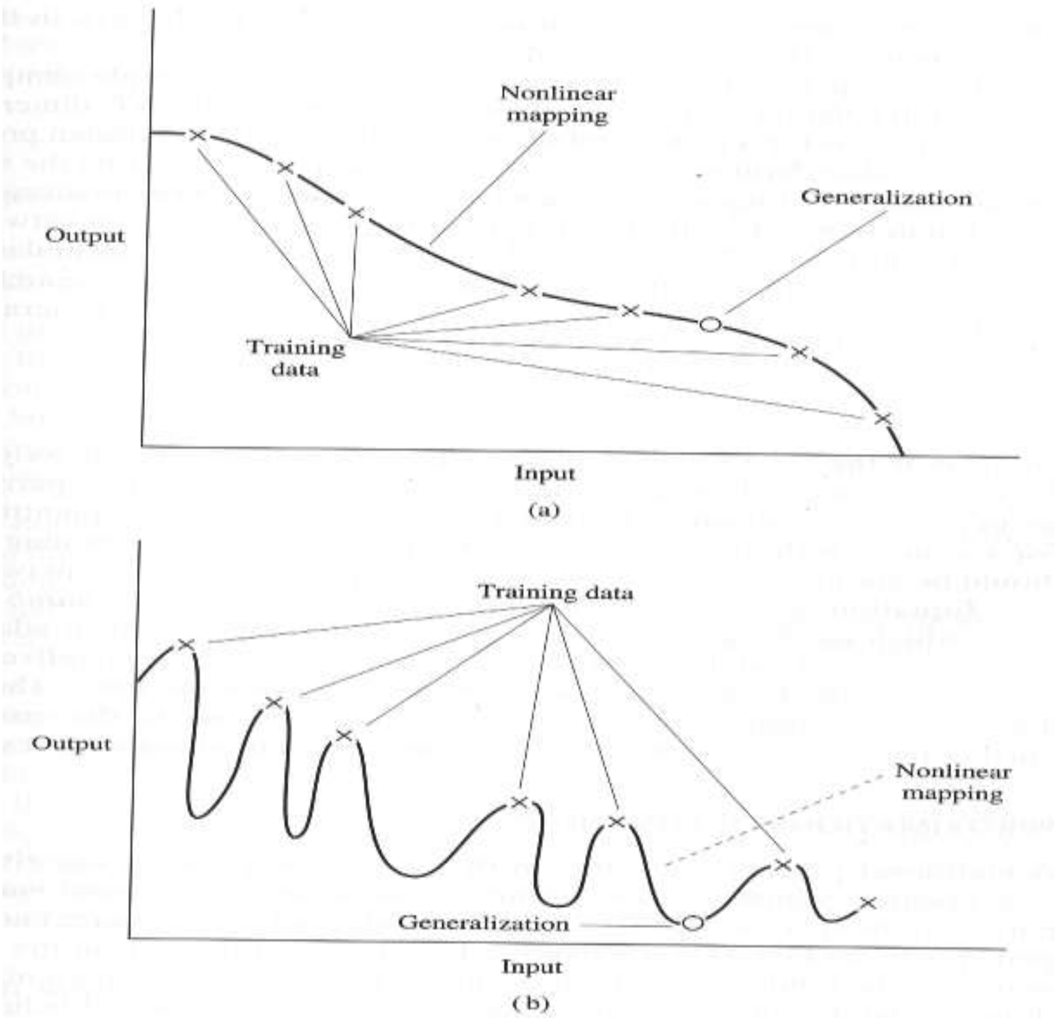


FIGURE 4.19 (a) Properly fitted data (good generalization) (b) Overfitted data (poor generalization).

Expressive capabilities of NN

Boolean functions:

- Every boolean function can be represented by network with single hidden layer
- but might require exponential hidden units

Continuous functions:

- Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer
- Any function can be approximated with arbitrary accuracy by a network with two hidden layers

Generalized Delta Rule

- If η small $\Rightarrow$ Slow rate of learning

  If η large $\Rightarrow$ Large changes of weights

  $\Rightarrow$ NN can become unstable (oscillatory)

- Method to overcome above drawback:

  include a momentum term in the delta

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

Generalized delta function

momentum constant

Generalized delta rule

- the momentum accelerates the descent in steady downhill directions.
- the momentum has a stabilizing effect in directions that oscillate in time.

η adaptation

Heuristics for accelerating the convergence of the back-prop algorithm through η adaptation:

- Heuristic 1: Every weight should have its own η.

- Heuristic 2: Every η should be allowed to vary from one iteration to the next.

NN DESIGN

- Data representation
- Network Topology
- Network Parameters
- Training
- Validation

Setting the parameters

- How are the weights initialised?
- How is the learning rate chosen?
- How many hidden layers and how many neurons?
- Which activation function ?
- How to preprocess the data ?
- How many examples in the training data set?

Some heuristics (1)

- Sequential x Batch algorithms: the sequential mode (pattern by pattern) is computationally faster than the batch mode (epoch by epoch)

Some heuristics (2)

- Maximization of information content: every training example presented to the backpropagation algorithm must maximize the information content.
  - The use of an example that results in the largest training error.
  - The use of an example that is radically different from all those previously used.

Some heuristics (3)

- Activation function: network learns faster with antisymmetric functions when compared to nonsymmetric functions.

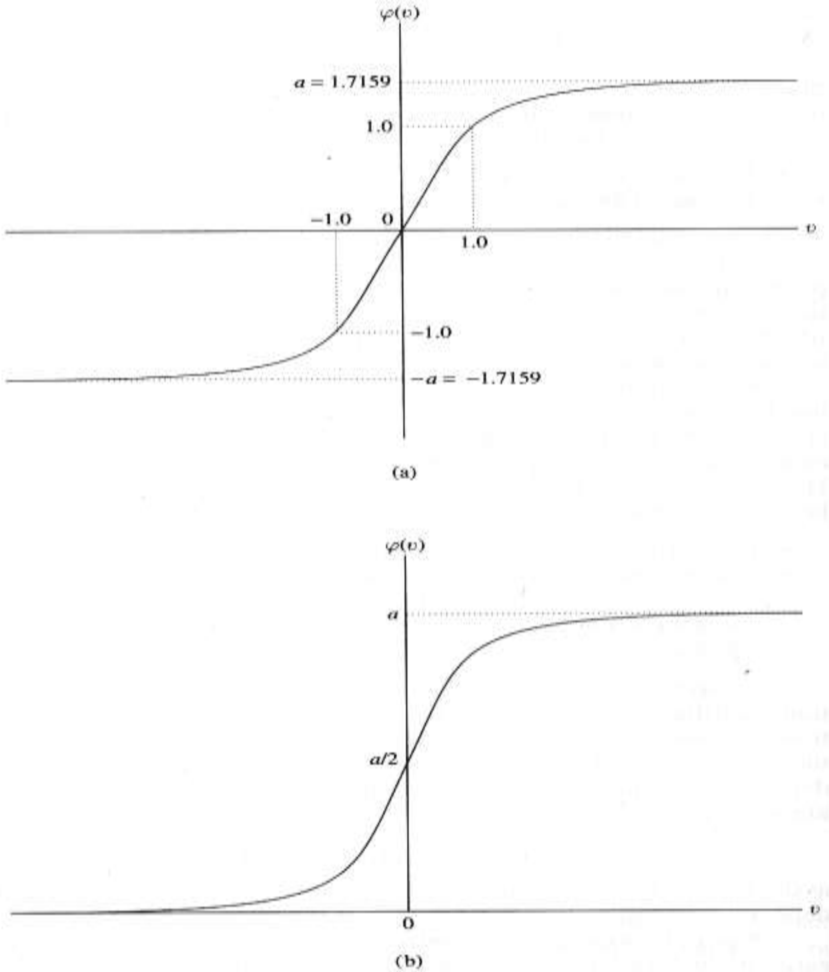
$$\varphi(\mathrm{v}) = \frac{1}{1+e^{-av}}$$

Sigmoidal function is nonsymmetric

$$\varphi(\mathrm{v}) = a\tanh(bv)$$

Hyperbolic tangent function is nonsymmetric

Some heuristics (3)



(a)

(b)

FIGURE 4.10 Antisymmetric activation function. (b) Nonsymmetric activation function.

Some heuristics (4)

- Target values: target values must be chosen within the range of the sigmoidal activation function.
- Otherwise, hidden neurons can be driven into saturation which slows down learning

Some heuristics (4)

- For the antisymmetric activation function it is necessary to design Є
- For a+: $d_j = a - \varepsilon$
- For –a: $d_j = -a + \varepsilon$
- If a=1.7159 we can set Є=0.7159 then d=±1

Some heuristics (5)

- Inputs normalisation:
  - Each input variable should be processed so that the mean value is small or close to zero or at least very small when compared to the standard deviation.
  - Input variables should be uncorrelated.
  - Decorrelated input variables should be scaled so their covariances are approximately equal.

Some heuristics (5)

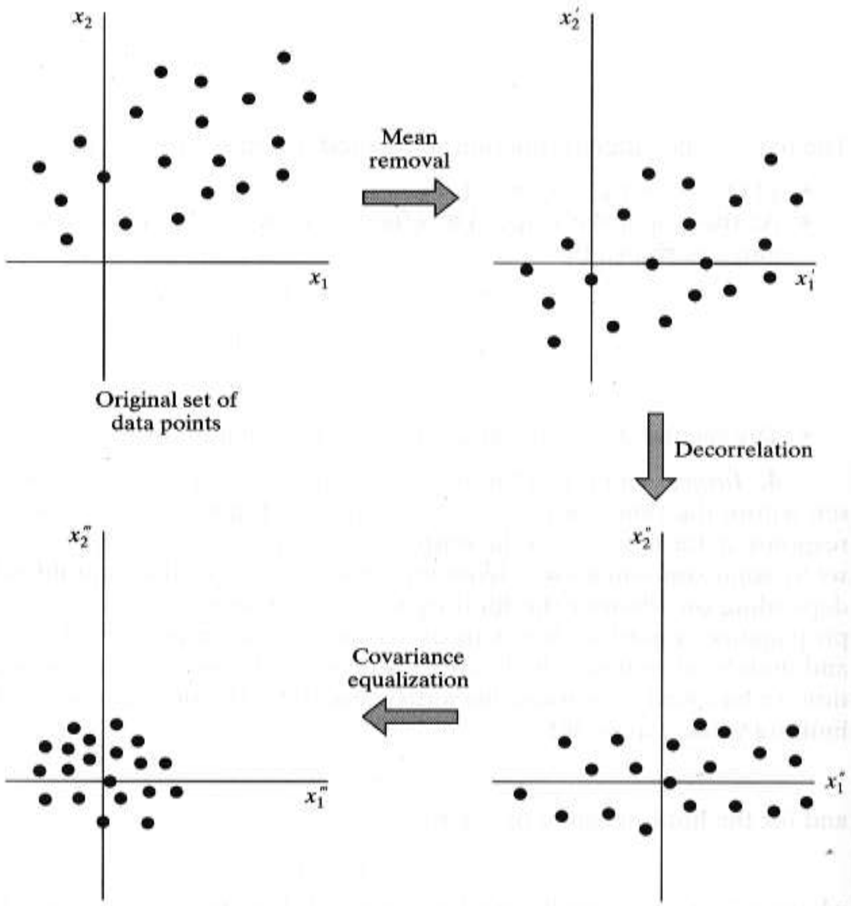


FIGURE 4.11 Illustrating the operation of mean removal, decorrelation, and covariance equalization for a two-dimensional input space.

Some heuristics (6)

- Initialisation of weights:
  - If synaptic weights are assigned large initial values neurons are driven into saturation. Local gradients become small so learning rate becomes small.
  - If synaptic weights are assigned small initial values algorithms operate around the origin. For the hyperbolic activation function the origin is a saddle point.

Some heuristics (6)

- Weights must be initialised for the standard deviation of the local induced field v lies in the transition between the linear and saturated parts.

$$\sigma_v = 1$$

$$\sigma_w = m^{-1/2}$$ m=number of weights

Some heuristics (7)

- Learning rate:
  - The right value of η depends on the application. Values between 0.1 and 0.9 have been used in many applications.
  - Other heuristics adapt η during the training as described in previous slides.

Some heuristics (8)

- How many layers and neurons
  - The number of layers and of neurons depend on the specific task. In practice this issue is solved by trial and error.
  - Two types of adaptive algorithms can be used:
    - **start from a large network and successively remove some neurons and links until network performance degrades.**
    - **begin with a small network and introduce new neurons until performance is satisfactory.**

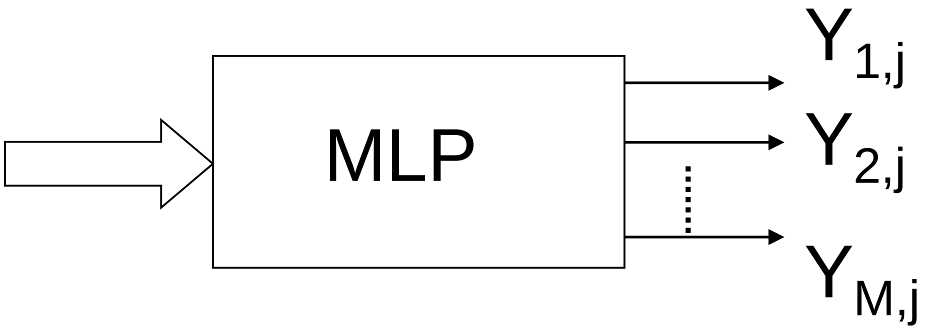
Some heuristics (9)

- How many training data ?
  - Rule of thumb: the number of training examples should be at least five to ten times the number of weights of the network.

Output representation and decision rule

- M-class classification problem

$Y_{k,j}(x_j)=F_k(x_j)$, k=1,...,M



Data representation

$$d_{k,j} = \begin{cases} 1, x_j \in C_k \\ 0, x_j \notin C_k \end{cases}$$

$$\begin{array}{cl} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} & \begin{array}{l} \\ \\ \leftarrow \text{ Kth element} \\ \\ \\ \end{array} \end{array}$$

MLP and the a posteriori class probability

- **A multilayer perceptron classifier (using the logistic function) aproximate the a posteriori class probabilities, provided that the size of the training set is large enough.**

The Bayes rule

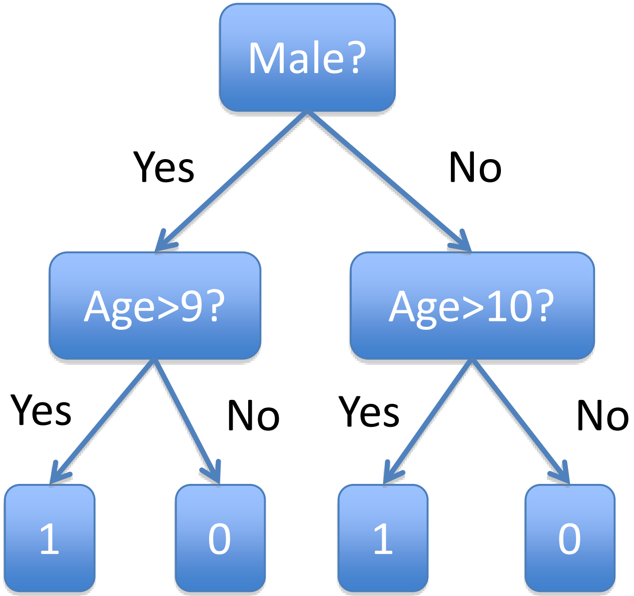
- An appropriate output decision rule is the (approximate) Bayes rule generated by the a ***posteriori*** probability estimates:
- xЄC$_k$ if F$_k$(x)>F$_j$(x) for all $j \neq k$

$$F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \\ \cdots \\ F_M(x) \end{bmatrix}$$

UNIT- V
An Introduction to Ensemble Methods
Bagging, Boosting, Random Forests, and More

Supervised Learning

- **Goal:** learn predictor h(x)
  - High accuracy (low error)
  - Using training data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$



Person	Age	Male?	Height > 55”	
Alice	14	0	1	✓
Bob	10	1	1	✓
Carol	13	0	1	✓
Dave	8	1	0	✓
Erin	11	0	0	✗
Frank	9	1	1	✗
Gena	8	0	0	✓

$$x = \begin{bmatrix} age \\ 1_{[gender=male]} \end{bmatrix} \qquad y = \begin{cases} 1 & height > 55'' \\ 0 & height \leq 55'' \end{cases}$$

Different Classifiers

- Performance
  - None of the classifiers is perfect
  - Complementary
    - Examples which are not correctly classified by one classifier may be correctly classified by the other classifiers
- Potential Improvements?
  - Utilize the complementary property

Ensembles of Classifiers

- Idea
  - Combine the classifiers to improve the performance
- Ensembles of Classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting

Example: Weather Forecast

Reality	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈ X	☀	☀ X	⛈	☀	⛈ X
2	☀ X	☀	☀	☀ X	⛈	☀	⛈ X
3	⛈	☀	⛈ X	⛈	☀ X	⛈ X	☀
4	⛈	☀	⛈ X	⛈	☀ X	☀	☀
5	⛈	⛈ X	☀	⛈	⛈	⛈ X	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Outline

- **Bias/Variance Tradeoff**

- Ensemble methods that minimize variance
  - Bagging
  - Random Forests

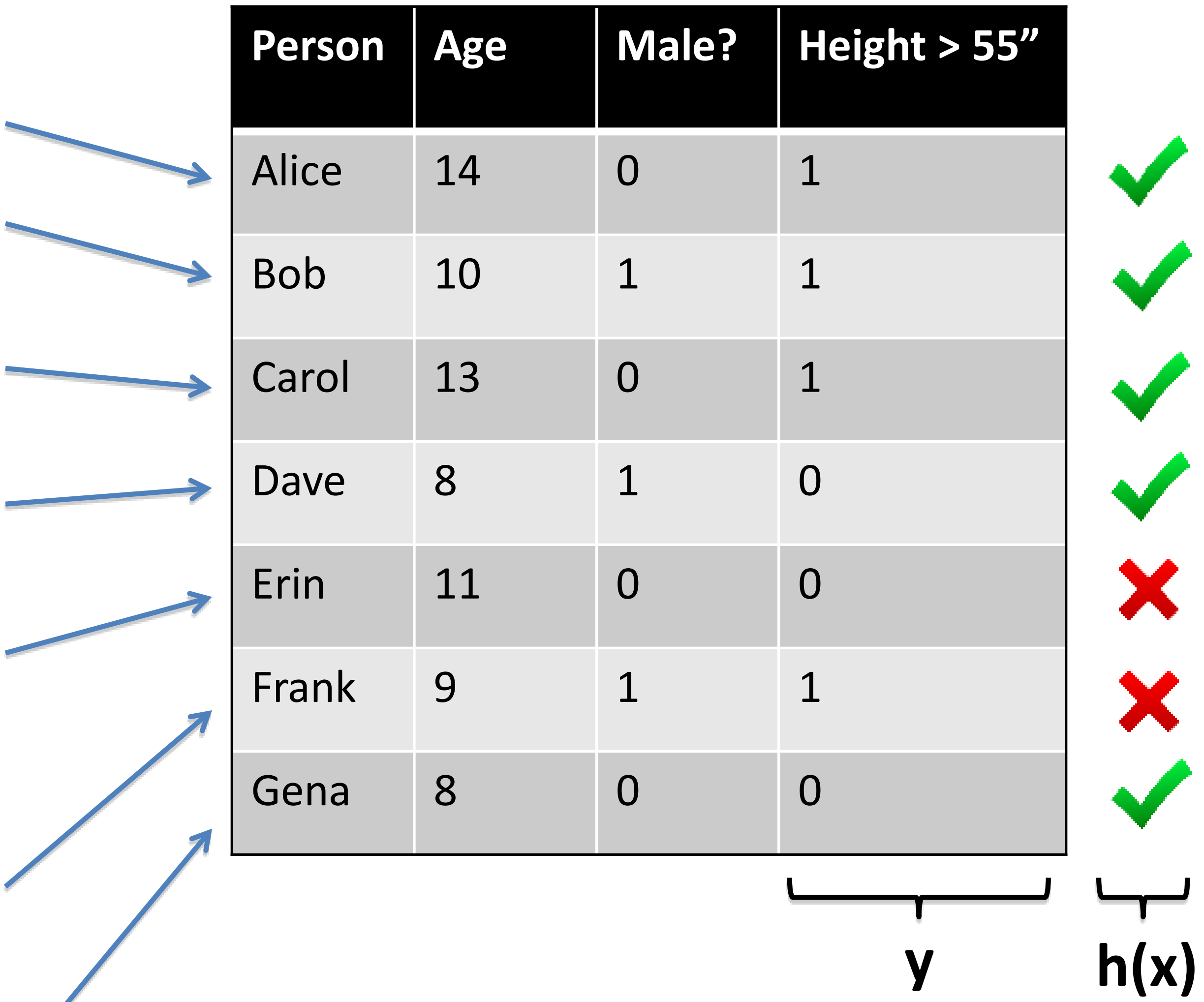
- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Generalization Error

- **“True” distribution:** $P(x,y)$
  - Unknown to us
- **Train:** $h(x) = y$
  - Using training data $S = \{(x_1,y_1),\ldots,(x_n,y_n)\}$
  - Sampled from $P(x,y)$
- **Generalization Error:**
  - $\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\ f(h(x),y)\ ]$
  - E.g., $f(a,b) = (a-b)^2$

Person	Age	Male?	Height > 55”
James	11	1	1
Jessica	14	0	1
Alice	14	0	1
Amy	12	0	1
Bob	10	1	1
Xavier	9	1	0
Cathy	9	0	1
Carol	13	0	1
Eugene	13	1	0
Rafael	12	1	1
Dave	8	1	0
Peter	9	1	0
Henry	13	1	0
Erin	11	0	0
Rose	7	0	0
Iain	8	1	1
Paulo	12	1	0
Margaret	10	0	1
Frank	9	1	1
Jill	13	0	0
Leon	10	1	0
Sarah	12	0	0
Gena	8	0	0
Patrick	5	1	1

⋮



Person	Age	Male?	Height > 55”	
Alice	14	0	1	✓
Bob	10	1	1	✓
Carol	13	0	1	✓
Dave	8	1	0	✓
Erin	11	0	0	✗
Frank	9	1	1	✗
Gena	8	0	0	✓

Generalization Error:

$$\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\, f(h(x),y)\,]$$

Bias/Variance Tradeoff

- Treat h(x|S) has a random function
  - Depends on training data S

- $\mathcal{L} = E_S[\ E_{(x,y)\sim P(x,y)}[\ f(h(x|S),y)\ ]\ ]$
  - Expected generalization error
  - Over the randomness of S

Bias/Variance Tradeoff

- Squared loss: $f(a,b) = (a-b)^2$
- Consider one data point (x,y)
- Notation:
  - $Z = h(x|S) - y$
  - $\check{z} = E_S[Z]$
  - $Z-\check{z} = h(x|S) - E_S[h(x|S)]$

$$\boxed{\begin{aligned} E_S[(Z-\check{z})^2] &= E_S[Z^2 - 2Z\check{z} + \check{z}^2] \\ &= E_S[Z^2] - 2E_S[Z]\check{z} + \check{z}^2 \\ &= E_S[Z^2] - \check{z}^2 \end{aligned}}$$

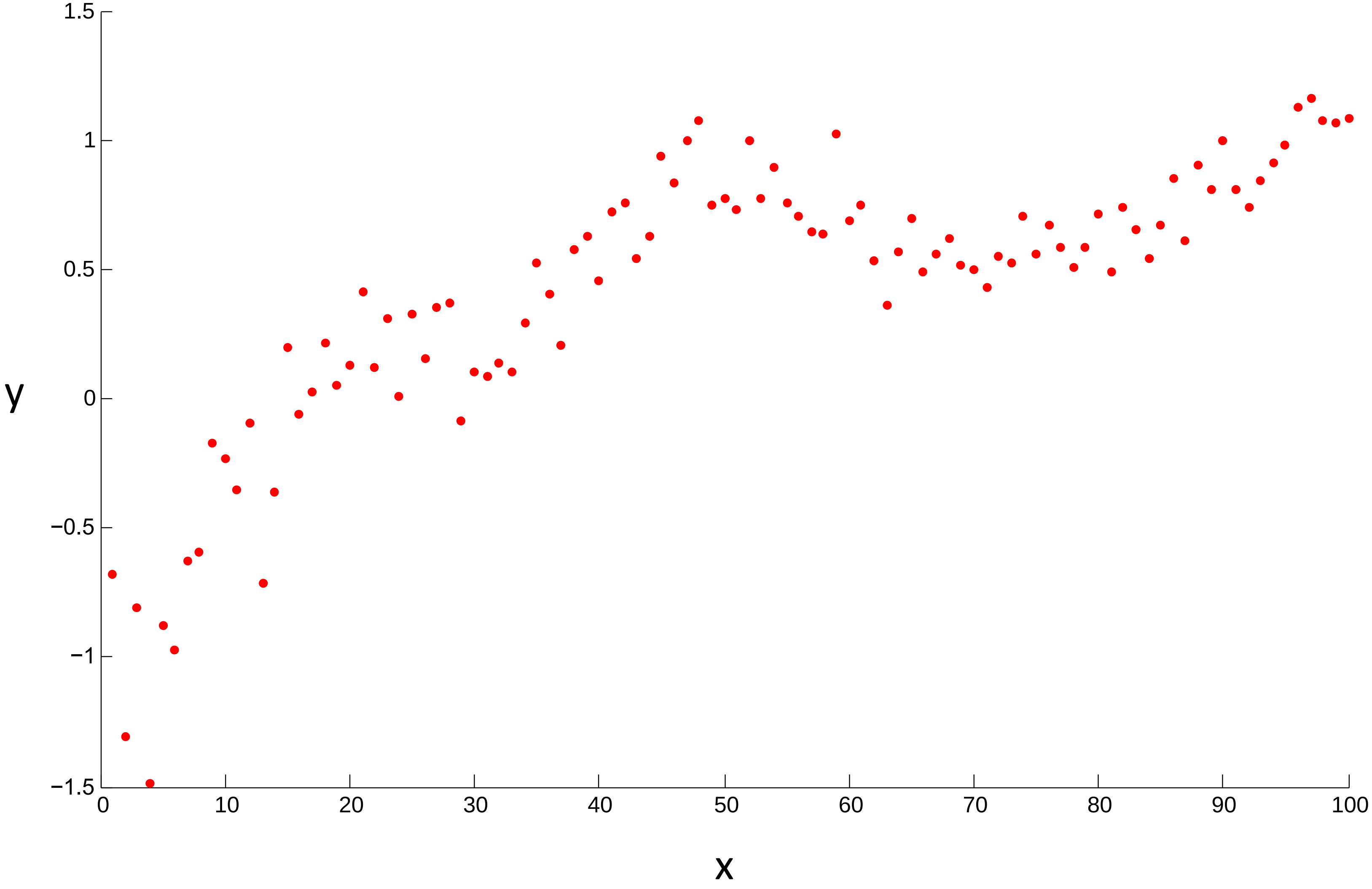
$$\boxed{\begin{aligned} \overbrace{E_S[f(h(x|S),y)]}^{\text{Expected Error}} &= E_S[Z^2] \\ &= \underbrace{E_S[(Z-\check{z})^2]}_{\text{Variance}} + \underbrace{\check{z}^2}_{\text{Bias}} \end{aligned}}$$

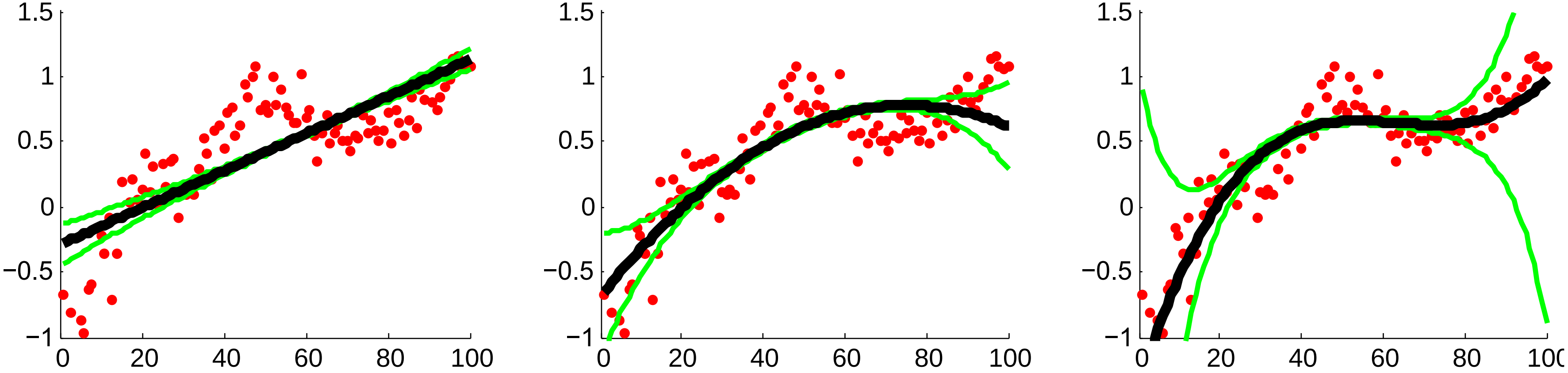
Bias/Variance for all (x,y) is expectation over P(x,y).

Can also incorporate measurement noise.

(Similar flavor of analysis for other loss functions.)

Example




1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
10

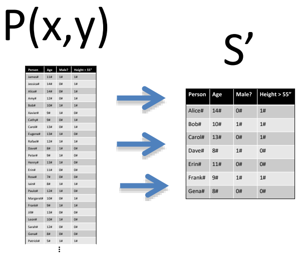
Outline

- Bias/Variance Tradeoff

- **Ensemble methods that minimize variance**
  - **Bagging**
  - **Random Forests**

- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Bagging



P(x,y)

Person	Age	Male?	Height > 55"
James#	11#	1#	1#
Jessica#	14#	0#	1#
Alice#	14#	0#	1#
Amy#	12#	0#	1#
Bob#	10#	1#	1#
Xavier#	9#	1#	0#
Cathy#	9#	0#	1#
Carol#	13#	0#	1#
Eugene#	13#	1#	0#
Rafael#	12#	1#	1#
Dave#	8#	1#	0#
Peter#	9#	1#	0#
Henry#	13#	1#	0#
Erin#	11#	0#	0#
Rose#	7#	0#	0#
Iain#	8#	1#	1#
Paulo#	12#	1#	0#
Margaret#	10#	0#	1#
Frank#	9#	1#	1#
Jill#	13#	0#	0#
Leon#	10#	1#	0#
Sarah#	12#	0#	0#
Gena#	8#	0#	0#
Patrick#	5#	1#	1#

⋮

S'

Person	Age	Male?	Height > 55"
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

- **Goal:** reduce variance

sampled independently

- **Ideal setting:** many training sets S'
  - Train model using each S'
  - Average predictions

Variance reduces linearly
Bias unchanged

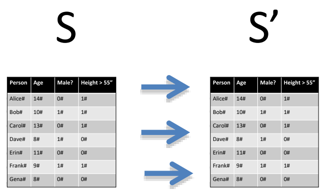
$$\boxed{\underbrace{E_S[(h(x|S) - y)^2]}_{\text{Expected Error}} = \underset{\substack{\uparrow\\ \textbf{Variance}}}{E_S[(Z-\check{z})^2]} + \underset{\substack{\uparrow\\ \textbf{Bias}}}{\check{z}^2}}$$

$$Z = h(x|S) - y$$
$$\check{z} = E_S[Z]$$

"Bagging Predictors" [Leo Breiman, 1994]
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

Bagging

- **Goal:** reduce variance



S

Person	Age	Male?	Height > 55"
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

S'

Person	Age	Male?	Height > 55"
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

from S

- **In practice:** resample S' with replacement
  - – Train model using each S'
  - – Average predictions

Variance reduces sub-linearly
(Because S' are correlated)
Bias often increases slightly

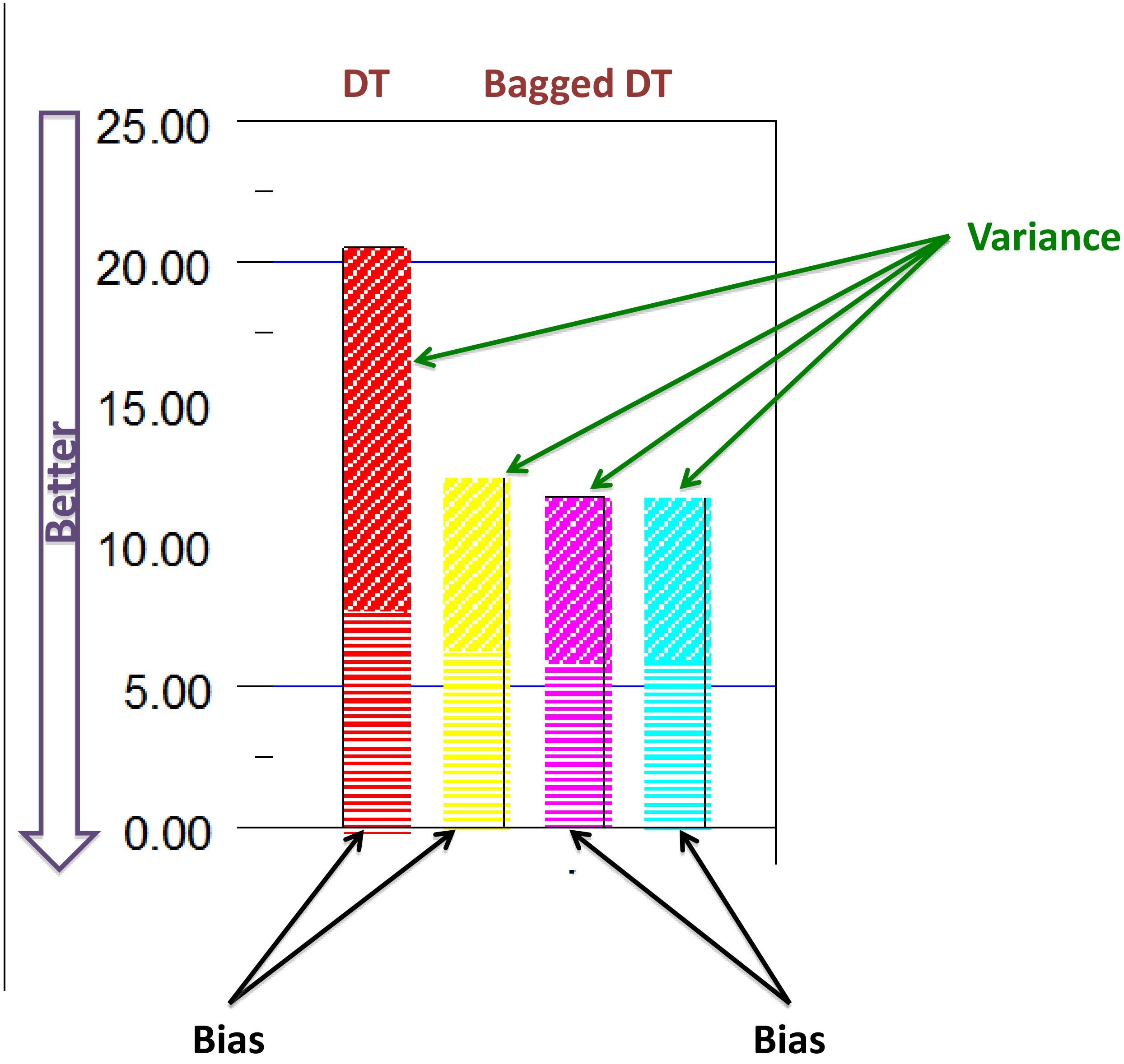
$$\boxed{E_S[(h(x|S) - y)^2] = E_S[(Z-\check{z})^2] + \check{z}^2}$$

$$\underbrace{\qquad\qquad\qquad}_{\text{Expected Error}} \qquad \underset{\textbf{Variance}}{\uparrow} \qquad \underset{\textbf{Bias}}{\uparrow}$$

$$Z = h(x|S) - y$$
$$\check{z} = E_S[Z]$$

"Bagging Predictors" [Leo Breiman, 1994]
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

Bagging = Bootstrap Aggregation



“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

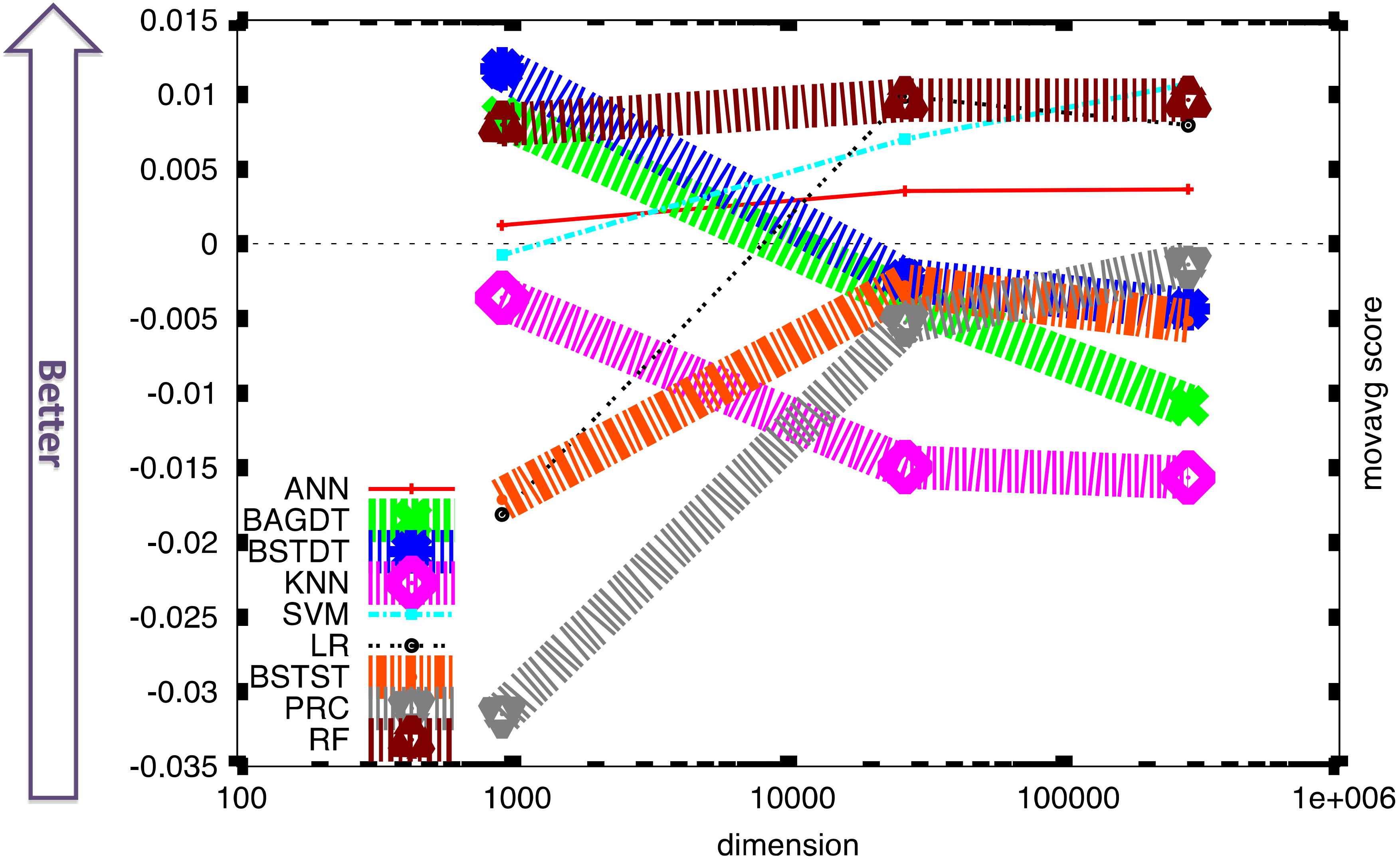
Random Forests

- **Goal:** reduce variance
  - Bagging can only do so much
  - Resampling training data asymptotes

- **Random Forests:** sample data & features!
  - Sample S’
  - Train DT
    - At each node, sample features (sqrt)
  - Average predictions

Further de-correlates trees

“Random Forests – Random Features” [Leo Breiman, 1997]
http://oz.berkeley.edu/~breiman/random-forests.pdf



Average performance over many datasets
Random Forests perform the best

"An Empirical Evaluation of Supervised Learning in High Dimensions"
Caruana, Karampatziakis & Yessenalina, ICML 2008

Structured Random Forests

- DTs normally train on unary labels y=0/1
- What about structured labels?
  - Must define information gain of structured labels
- Edge detection:
  - E.g., structured label is a 16x16 image patch
  - Map structured labels to another space
    - where entropy is well defined

“Structured Random Forests for Fast Edge Detection”
Dollár & Zitnick, ICCV 2013

Outline

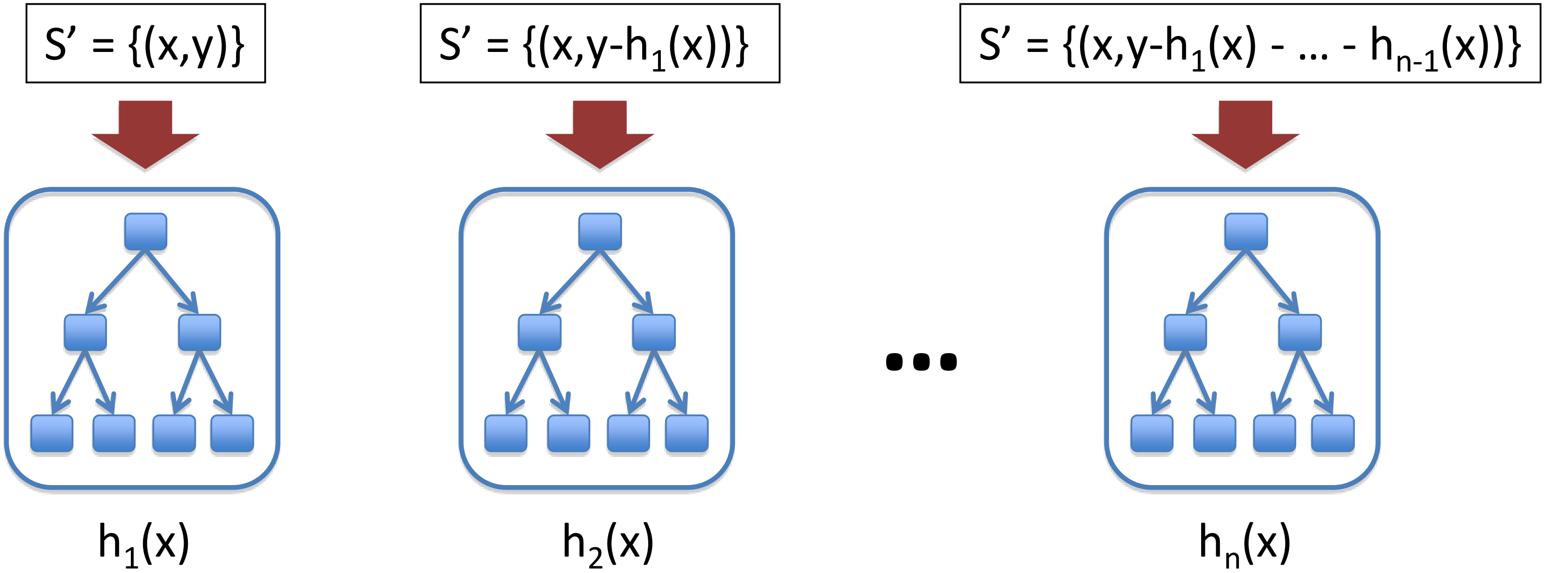
- Bias/Variance Tradeoff

- Ensemble methods that minimize variance
  - Bagging
  - Random Forests

- **Ensemble methods that minimize bias**
  - **Functional Gradient Descent**
  - **Boosting**
  - **Ensemble Selection**

Functional Gradient Descent

$$h(x) = h_1(x) + h_2(x) + \dots + h_n(x)$$



$S' = \{(x,y)\}$

$S' = \{(x,y-h_1(x))\}$

$S' = \{(x,y-h_1(x) - \dots - h_{n-1}(x))\}$

$h_1(x)$ $h_2(x)$ … $h_n(x)$

http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

Coordinate Gradient Descent

- Learn w so that $h(x) = w^T x$

- Coordinate descent
  - Init w = 0
  - Choose dimension with highest gain
    - Set component of w
  - Repeat

Coordinate Gradient Descent

- Learn w so that h(x) = $w^T x$
- Coordinate descent
  - Init w = 0
  - Choose dimension with highest gain
    - Set component of w
  - Repeat

$$w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Coordinate Gradient Descent

- Learn w so that h(x) = $w^T x$
- Coordinate descent
  - Init w = 0
  - Choose dimension with highest gain
    - Set component of w
  - Repeat

$$w = \begin{bmatrix} 0 \\ 0 \\ +3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Coordinate Gradient Descent

- Learn w so that h(x) = $w^T x$

- Coordinate descent
  - Init w = 0
  - Choose dimension with highest gain
    - Set component of w
  - Repeat

$$w = \begin{bmatrix} 0 \\ 0 \\ +3 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1.5 \\ 0 \\ 0 \end{bmatrix}$$

Coordinate Gradient Descent

- Learn w so that h(x) = $w^T x$
- Coordinate descent
  - – Init w = 0
  - – Choose dimension with highest gain
    - Set component of w
  - – Repeat

$$w = \begin{bmatrix} +2.1 \\ 0 \\ +3 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1.5 \\ 0 \\ 0 \end{bmatrix}$$

Coordinate Gradient Descent

- Learn w so that h(x) = $w^T x$

- Coordinate descent
  - – Init w = 0
  - – Choose dimension with highest gain
    - Set component of w
  - – Repeat

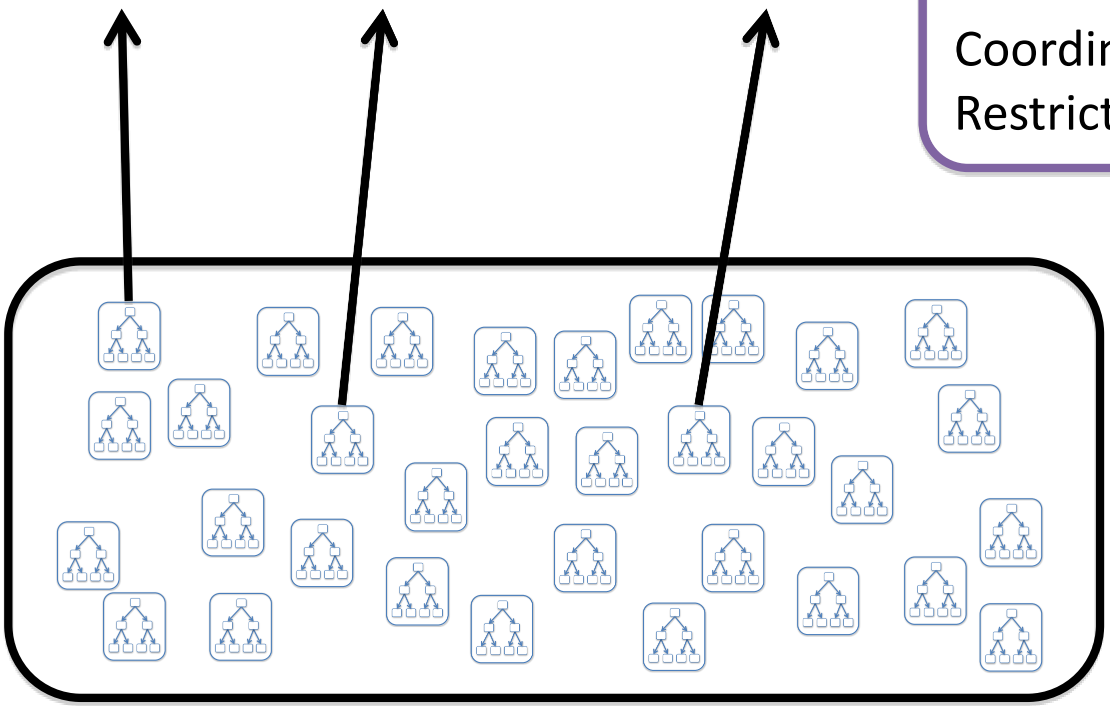
$$w = \begin{bmatrix} +2.1 \\ 0 \\ +3 \\ 0 \\ 0 \\ -0.9 \\ 0 \\ -1.5 \\ 0 \\ 0 \end{bmatrix}$$

Functional Gradient Descent

$h(x) = h_1(x) + h_2(x) + \ldots + h_n(x)$

Coordinate descent in function space

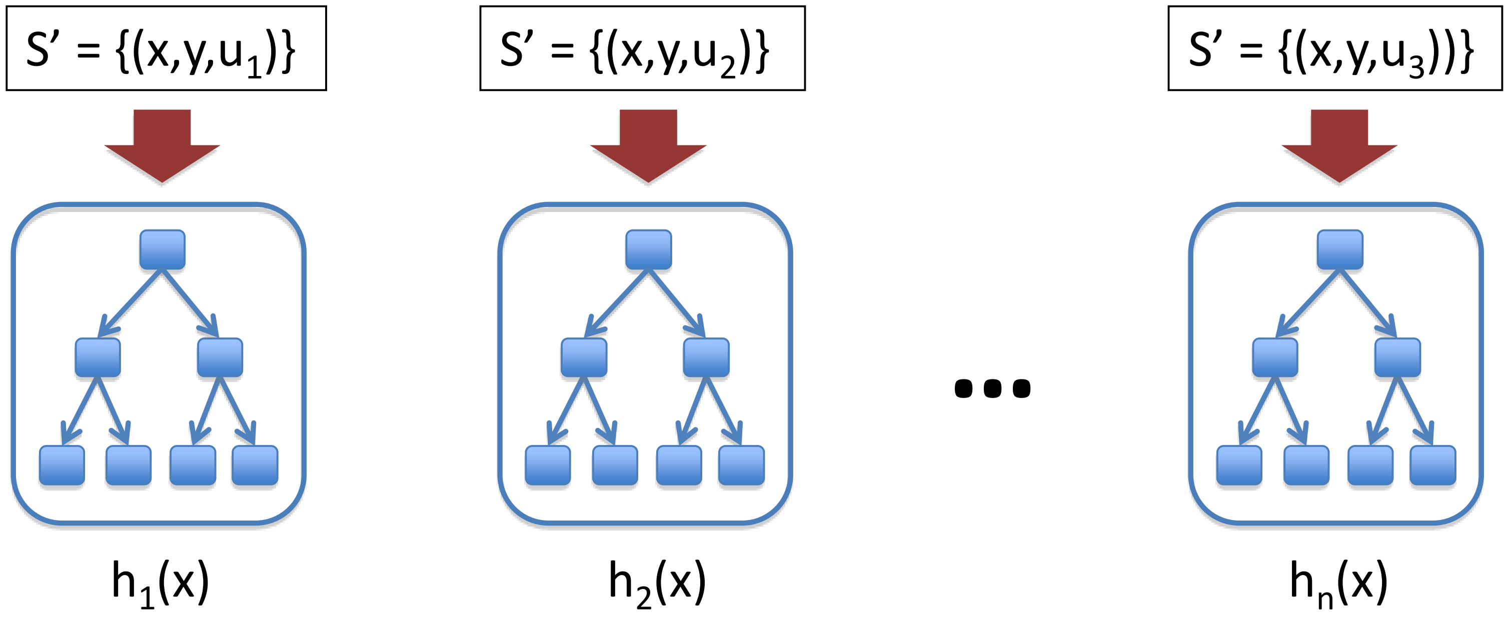
Restrict weights to be 0,1,2,…



“Function Space”
(All possible DTs)

Boosting (AdaBoost)

$$h(x) = a_1h_1(x) + a_2h_2(x) + \dots + a_3h_n(x)$$



$S' = \{(x,y,u_1)\}$

$S' = \{(x,y,u_2)\}$

$S' = \{(x,y,u_3))\}$

$h_1(x)$

$h_2(x)$

…

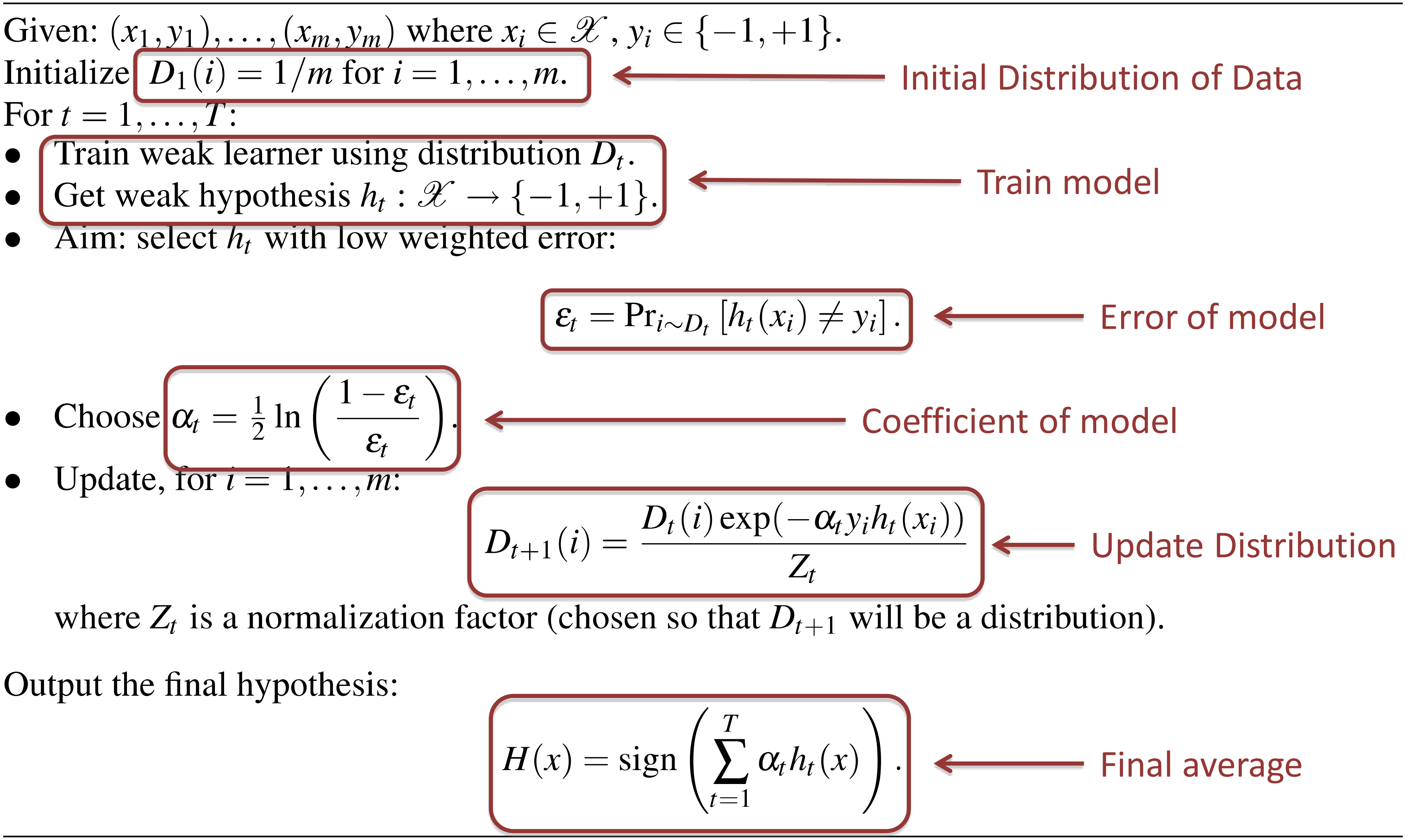
$h_n(x)$

u – weighting on data points

a – weight of linear combination

https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf

Stop when validation performance plateaus (will discuss later)



Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.

Initialize $\boxed{D_1(i) = 1/m \text{ for } i = 1, \ldots, m.}$ ← Initial Distribution of Data

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.

← Train model

- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].}$$

← Error of model

- Choose $\boxed{\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)}$. ← Coefficient of model
- Update, for $i = 1, \ldots, m$:

$$\boxed{D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}}$$

← Update Distribution

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

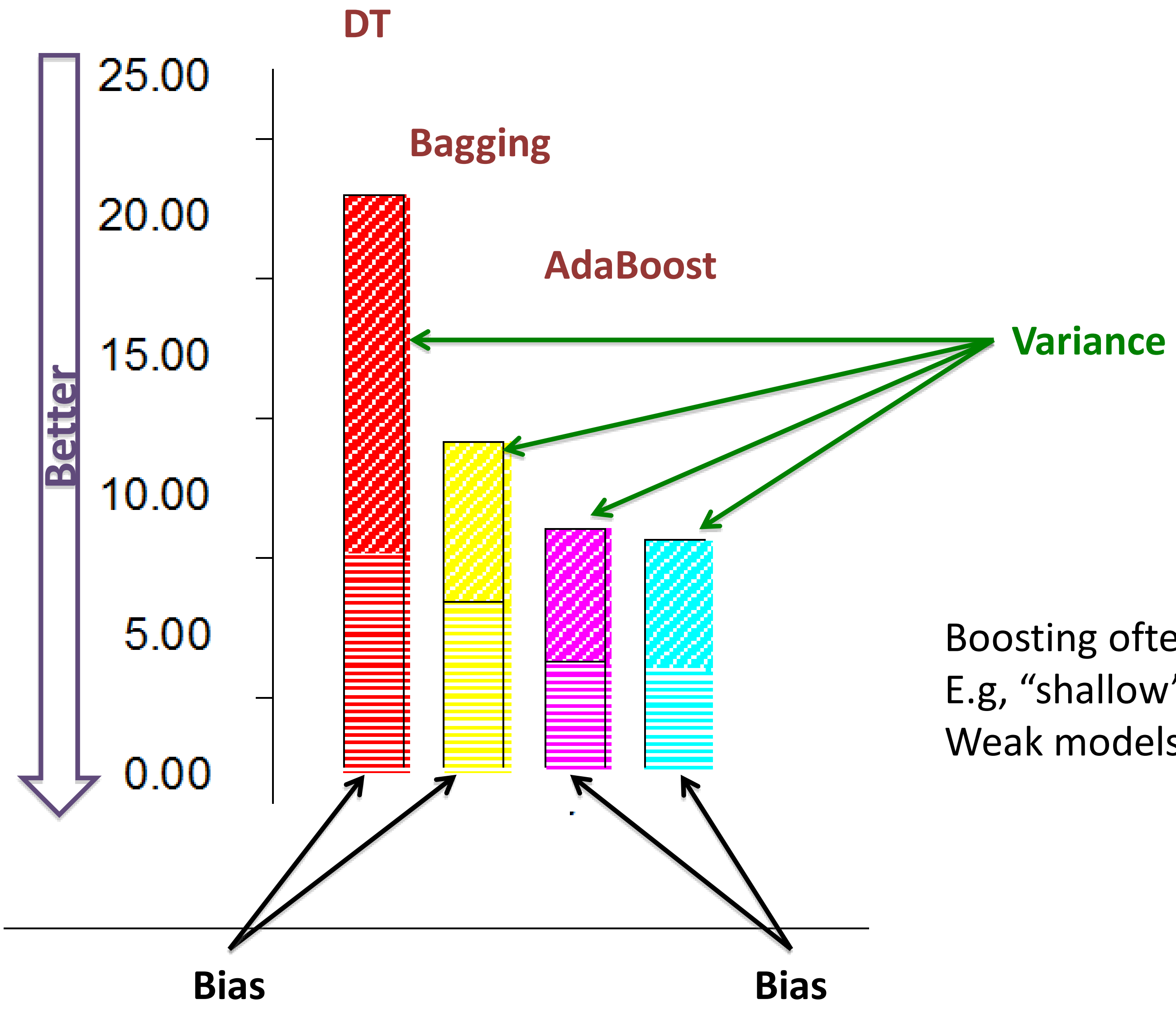
Output the final hypothesis:

$$\boxed{H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).}$$

← Final average

Theorem: training error drops exponentially fast

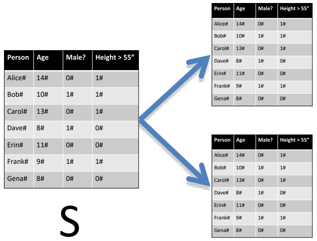
https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf



Boosting often uses weak models
E.g, “shallow” decision trees
Weak models have lower variance

“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

Ensemble Selection



Person	Age	Male?	Height > 55”
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

S

Person	Age	Male?	Height > 55”
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

Training S’

Person	Age	Male?	Height > 55”
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

Validation V’

H = {2000 models trained using S’}

Maintain ensemble model as combination of H:

$h(x) = h_1(x) + h_2(x) + \ldots + h_n(x) \; + h_{n+1}(x)$

Denote as h_{n+1}

Add model from H that maximizes performance on V’

Repeat

Models are trained on S’
Ensemble built to optimize V’

“Ensemble Selection from Libraries of Models”
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

Method	Minimize Bias?	Minimize Variance?	Other Comments
Bagging	Complex model class. (Deep DTs)	Bootstrap aggregation (resampling training data)	Does not work for simple models.
Random Forests	Complex model class. (Deep DTs)	Bootstrap aggregation + bootstrapping features	Only for decision trees.
Gradient Boosting (AdaBoost)	Optimize training performance.	Simple model class. (Shallow DTs)	Determines which model to add at run-time.
Ensemble Selection	Optimize validation performance	Optimize validation performance	Pre-specified dictionary of models learned on training set.

…and many other ensemble methods as well.

- State-of-the-art prediction performance
  - Won Netflix Challenge
  - Won numerous KDD Cups
  - Industry standard

References & Further Reading

“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants” Bauer & Kohavi, Machine Learning, 36, 105–139 (1999)

“Bagging Predictors” Leo Breiman, Tech Report #421, UC Berkeley, 1994, http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

“An Empirical Comparison of Supervised Learning Algorithms” Caruana & Niculescu-Mizil, ICML 2006

“An Empirical Evaluation of Supervised Learning in High Dimensions” Caruana, Karampatziakis & Yessenalina, ICML 2008

“Ensemble Methods in Machine Learning” Thomas Dietterich, *Multiple Classifier Systems*, 2000

“Ensemble Selection from Libraries of Models” Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

“Getting the Most Out of Ensemble Selection” Caruana, Munson, & Niculescu-Mizil, ICDM 2006

“Explaining AdaBoost” Rob Schapire, https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf

“Greedy Function Approximation: A Gradient Boosting Machine”, Jerome Friedman, 2001, http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

“Random Forests – Random Features” Leo Breiman, Tech Report #567, UC Berkeley, 1999,

“Structured Random Forests for Fast Edge Detection” Dollár & Zitnick, ICCV 2013

“ABC-Boost: Adaptive Base Class Boost for Multi-class Classification” Ping Li, ICML 2009

“Additive Groves of Regression Trees” Sorokina, Caruana & Riedewald, ECML 2007, http://additivegroves.net/

“Winning the KDD Cup Orange Challenge with Ensemble Selection”, Niculescu-Mizil et al., KDD 2009

“Lessons from the Netflix Prize Challenge” Bell & Koren, SIGKDD Exporations 9(2), 75—79, 2007