Nonparametric methods

Overview and applications

Pedram Khoshdani
Hamid Reza Zaheri

Outline

1. Introduction
2. Parametric vs. Nonparametric
3. Histogram
4. Kernel Density Estimation
5. KNN
6. Bayesian Nonparametric
7. Gaussian Process
8. Introduction to Decision Trees
9. Random Forest
10. Conclusion

Introduction

Two different meanings of the term “nonparametric” in statistics:

1. distribution free methods: makes no assumption about the distribution of data. opposite to parametric models in a sense that the latter makes this assumption and attempts to fit the parameters of the model to data (increasing the likelihood of data belonging to distribution)

2. use a parametric probability distribution model without fixing the complexity of the model. model can grow in size.

in other words, NonParametric methods are:

- By classical statistics: statistical modelling without parameters, e.g.,nearest neighbour methods.
- By Bayesian statistics: statistical modelling with an infinite number of parameters.(e.g.Gaussian Process)

in General, statistical modeling with variable or unbounded number of parameters

[http://nicta.com.au/__data/assets/pdf_file/0015/37113/buntine_ai12_041212.pdf]

Parametric vs. Nonparametric

- models in Parametric approach consist of finite set of parameters θ. given the parameters, predictions, x, are independent of the observed data.

$$P(x|D,\theta) = P(x|\theta)$$

θ provides all the information about the data

- complexity of the model is bounded. models is not flexible.

- in nonparametric approach, distribution of data is not defined by a finite set of parameters, we assume θ is infinite dimensional. consider them as functions (practically, functions are defined on vectors with infinite length)
- as we get more data, model can grow to capture all the useful features of the data, and this property makes the model flexible.
- Nonparametric model does not take a predetermined form but the model is constructed according to information derived from the data.

Histogram

Histogram is a way to estimate the distribution of data without assuming any particular shape for distribution (gaussian, beta, etc.).

- In case of continuous random variable, it can estimate the probability distribution.
- In the case of discrete data, it can represent normalized relative frequency. histogram shows the proportion of cases that fall into each of several categories.
- The total area of a histogram is always normalized to 1. to display a valid probability.(thus, it is a frequentist approach)

algorithm for constructing histograms(density estimation of a continuous RV):

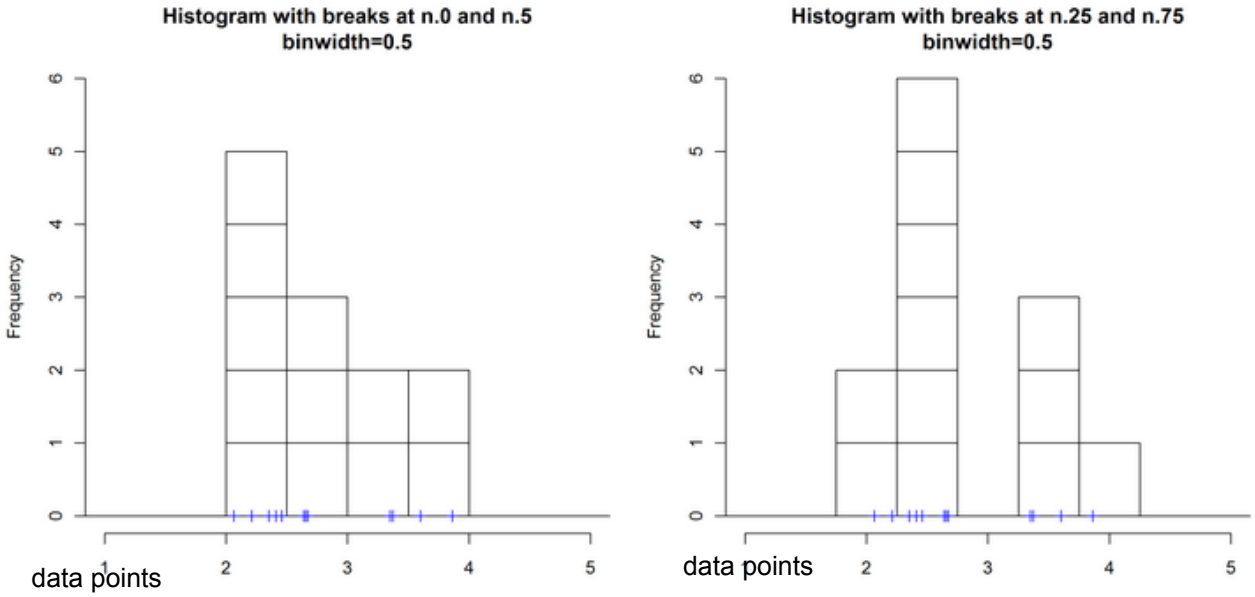
1. Span the data over the horizontal axis and divide them into equal intervals (bins)
2. Drop each value as a block of size 1 in the corresponding bin (data density).

Histogram

There are 2 parameters in histogram that affect its behavior:

- ➢ Size of the bin (binwidth)
- ➢ endpoints of the histogram

Effect of endpoints:

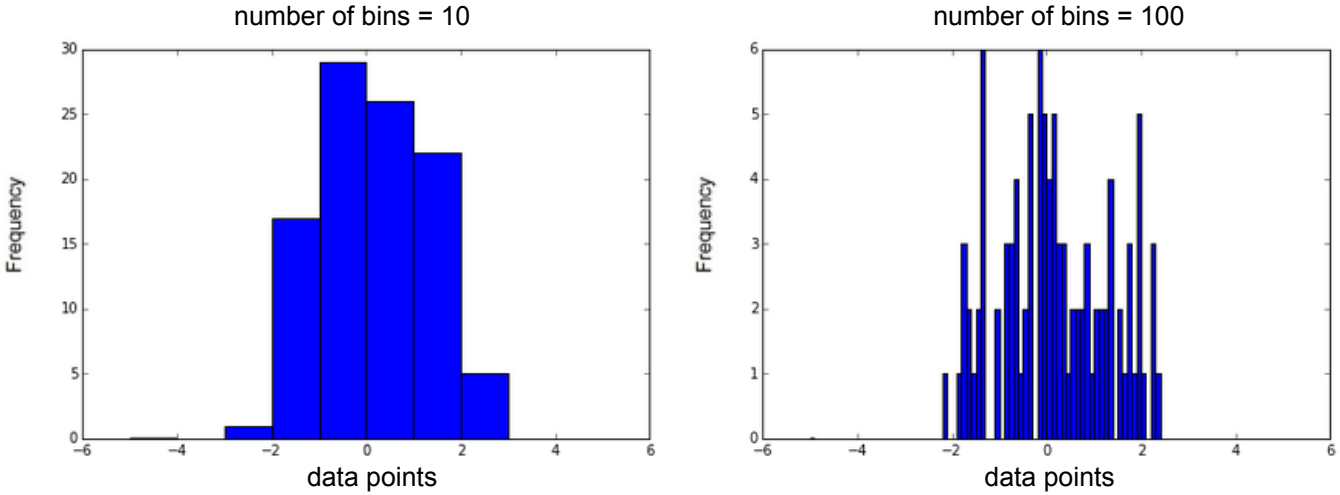


- In the left figure, endpoints are 0 and 0.5. it appears that the this density is unimodal and skewed to the right.
- In the right figure, endpoints are 0.25 and 0.75, We now have a completely different estimate of the density - it now appears to be bimodal.

[http://www.mvstat.net/tduong/research/seminars/seminar-2001-05/]

Histogram

Effect of the binwidth:



consider evaluating the probability of x= 3 or -3 in both histograms. the value in the right one is zero for both. but, in the left one, they have some other (and completely different) values.

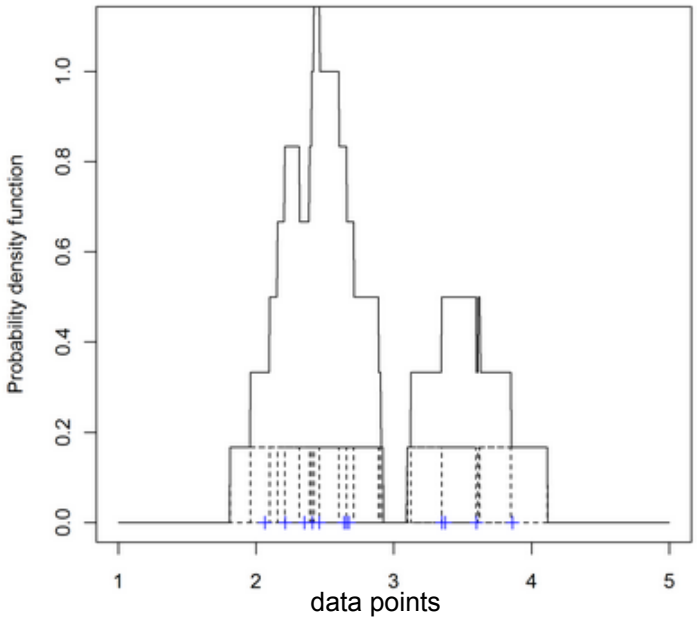
in Summary, histograms hold the following properties:

- Not smooth
- depend on endpoints of bins
- depend on width of bins

[http://www.mvstat.net/tduong/research/seminars/seminar-2001-05/]

Kernel Density Estimation

we can fix the problem of **endpoints** by centering each block at the corresponding data point. this is know as **box kernel density estimation.** (with boxcar kernel function). As a discontinuous kernel was used, the density is also still discontinuous.



In the figure, we place a block of width 1/2 and height 1/6 (the dotted boxes) as there are 12 data points, and then add them up. This density estimate (the solid curve) is less blocky, but, still not smooth.

http://www.mvstat.net/tduong/research/seminars/seminar-2001-05/

Kernel Density Estimation

solution to discontinuity problem in histogram: use smooth kernel for the building block to get a smooth density estimate.

*the problem of bandwidth (equivalent to histogram binwidth) still remains.

Kernel, is a non-negative function that integrates to one and has mean zero.

if we represent kernel by **K(.)** and bandwidth (smoothing parameter) by **h**(>0), the estimated density at any point x is:

$$\hat{f}_h = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

k_h =1/h K(x/h) is called the scaled kernel.

Kernel	K(u)
Uniform	$\frac{1}{2} I(\mid u \mid \leq 1)$
Triangle	$(1 - \mid u \mid)\ I(\mid u \mid \leq 1)$
Epanechnikov	$\frac{3}{4}(1 - u^2) I(\mid u \mid \leq 1)$
Quartic	$\frac{15}{16}(1 - u^2)^2 I(\mid u \mid \leq 1)$
Triweight	$\frac{35}{32}(1 - u^2)^3 I(\mid u \mid \leq 1)$
Gaussian	$\frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}u^2)$
Cosinus	$\frac{\pi}{4} cos(\frac{\pi}{2}u) I(\mid u \mid \leq 1)$

[http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/MISHRA/kde.html]

Kernel Density Estimation

Kernel Density Estimator (also called **Parzen window density estimator**) with **boxcar** (or Uniform) **kernel** function becomes:

$$P(x|D)=\frac{1}{2hn}\sum_{i=1}^{n} I(|x-x_i|<h)$$

where I(.) is an indicator function that takes on value 1 when the logical statement inside the parenthesis is true and 0 otherwise.

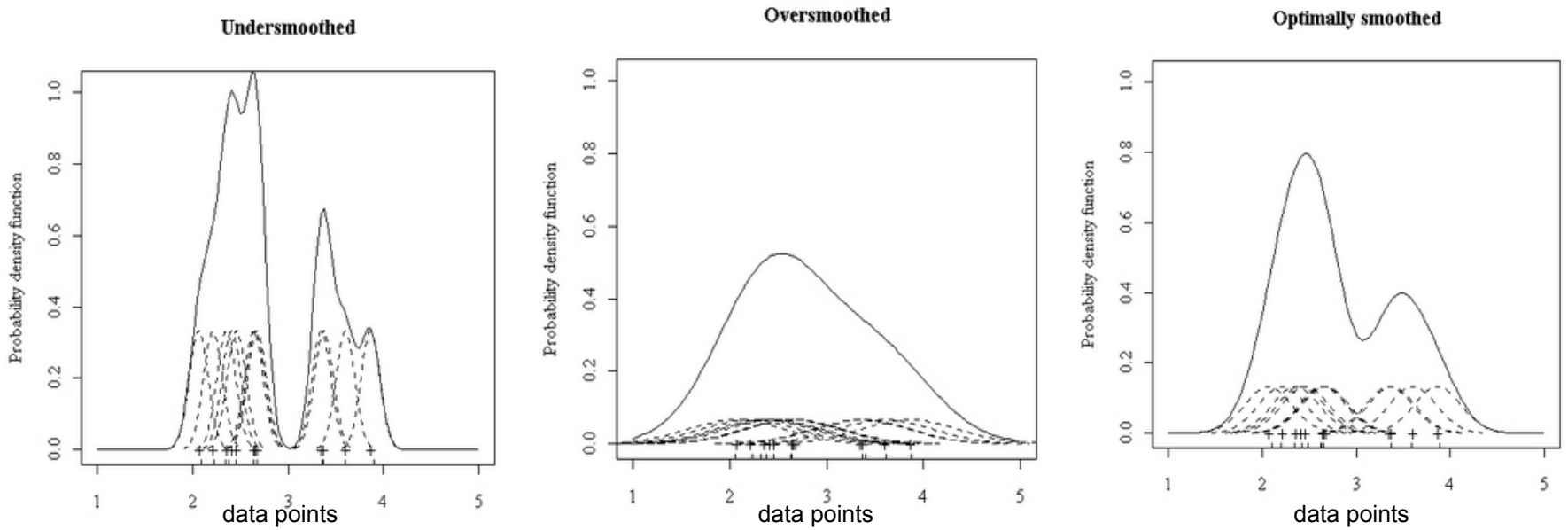
instead of boxcar kernel, one can uses the Gaussian kernel:

$$K_h=\frac{1}{\sqrt{2\pi}\,h}\sum_{i=1}^{n} exp(-\frac{x^2}{2h^2})$$

which results in smoother kernel.

Kernel Density Estimation

The quality of a kernel estimate depends less on the shape of the *K* than on the value of *its bandwidth h*. It's important to choose the most appropriate bandwidth as a value that is too small or too large is not useful. Small values of h lead to very spiky estimates (not much smoothing) while larger h values lead to oversmoothing.



[http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/MISHRA/kde.html]

Kernel Density Estimation

The bandwidth of the kernel is a free parameter which exhibits a strong influence on the resulting estimate. so, it should be wisely chosen.
A common way to do so is to use a bandwidth that minimizes the optimality criterion. The most common optimality criterion used to select this parameter is the expected L_2 risk function (also known as **mean integrated squared error**). The MISE of an estimate of an unknown probability density is given by:

$$MISE(h)=E\int(\hat{f}_h(x)-f(x))^2dx$$

where, f is the unknown density, f_h is its estimate based on a sample of n independent and identically distributed random variables. and E denotes the expected value with respect to that sample

under weak assumptions on f and K:

$$MISE(h)\simeq AMISE(h)=\frac{R(K)}{nh}+\frac{1}{4}m_2(K)^2h^4R(f'')$$

$$where\ \ R(g)=\int g(x)^2dx\ \ ,\ \ m_2(K)=\int x^2K(x)dx$$

f'' is the second derivative of f.

[wikipedia]

Kernel Density Estimation

The minimum of this AMISE is the solution to this differential equation

$$\frac{\partial}{\partial h} AMISE(h) = -\frac{R(K)}{nh^2} + m_2(K)^2 h^3 R(f'') = 0$$

$$\Rightarrow h_{AMISE} = \frac{R(K)^{1/5}}{m_2(K)^{2/5} R(f'')^{1/5} n^{1/5}}$$

In general, the AMISE still depends on the true underlying density (which is not available). So, Neither the AMISE nor the h_{AMISE} formulas are able to be used directly since they involve the unknown density function f or its second derivative f'', so a variety of automatic, data-based methods have been developed for selecting the bandwidth.This points out that the chosen bandwidth is an estimate of an asymptotic approximation

in conclusion, the properties of kernel density estimators are, as compared to histograms:

- smooth
- no end points
- depend on bandwidth

[wikipedia]

k-nearest neighbors algorithm

moving from KDE to KNN.
in brief, KDE:

- use regions centered at the data points
- allow the regions to overlap
- let each individual contribute a total density of 1/N
- with gaussian kernel, regions have soft edges to avoid discontinuity.

now, if we let the size of the region around each data point varies so that exactly K training data points fall inside the region, this will give us an algorithm called **K-Nearest Neighbors (KNN)**, which can be used for both classification and regression.

- KNN finds a predefined number of training samples closest in distance to the new point, and predict the label from these data points.
- The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning).
- The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.

k-nearest neighbors algorithm

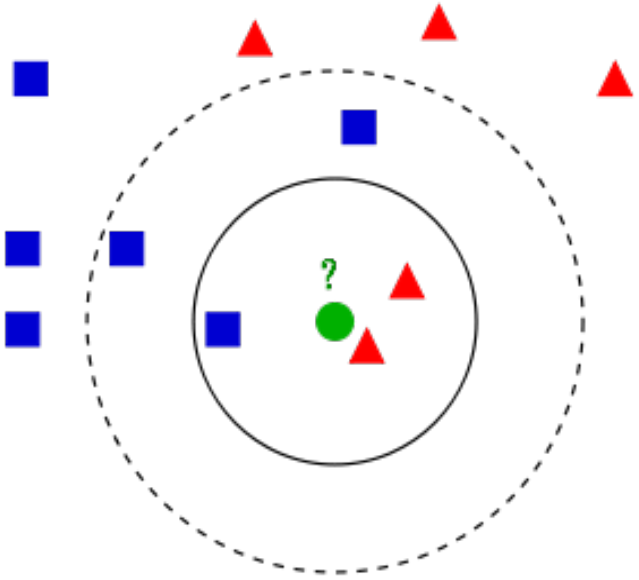
the input consists of the *k* closest training examples in the feature space. The output:

- for regression, is the average of the values of its *k* nearest neighbors.
- for classification, is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If*k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

Example:

cases:

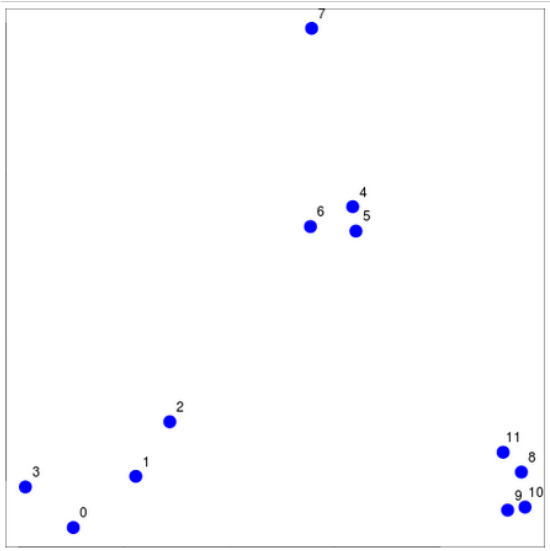
- If k = 3 (solid line circle)
- if k = 5 (dashed line circle)

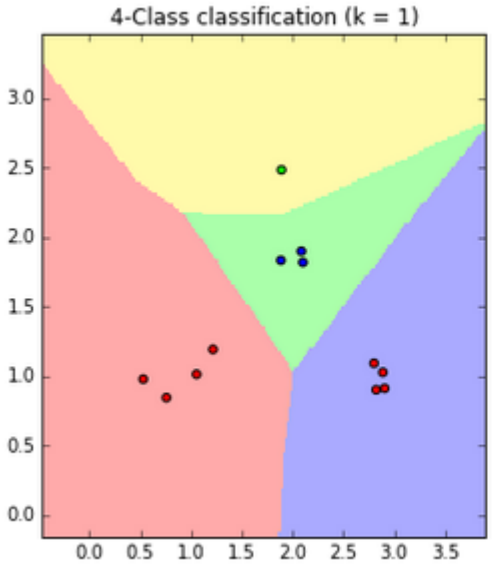


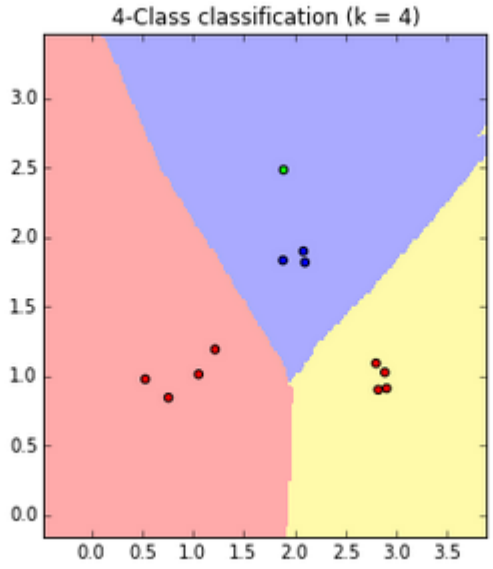
k-nearest neighbors algorithm

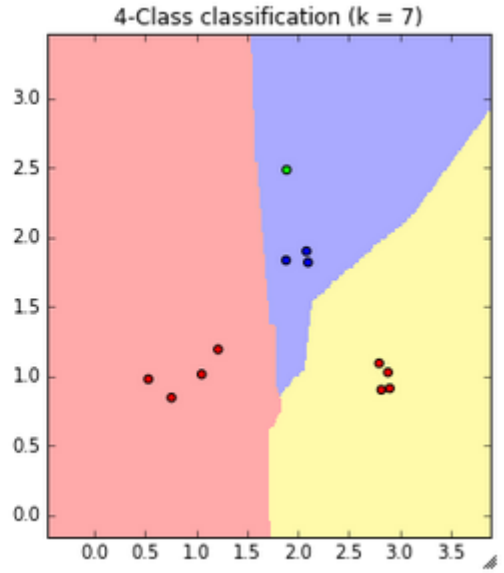
classification example:

12 data points with 4 different classes.
using different number of neighbors, here we can observe regions in the plane, where a test point would be assigned to any of the classes.
wights are uniform (contribution of all points in the neighborhood are the same, regardless of distance)









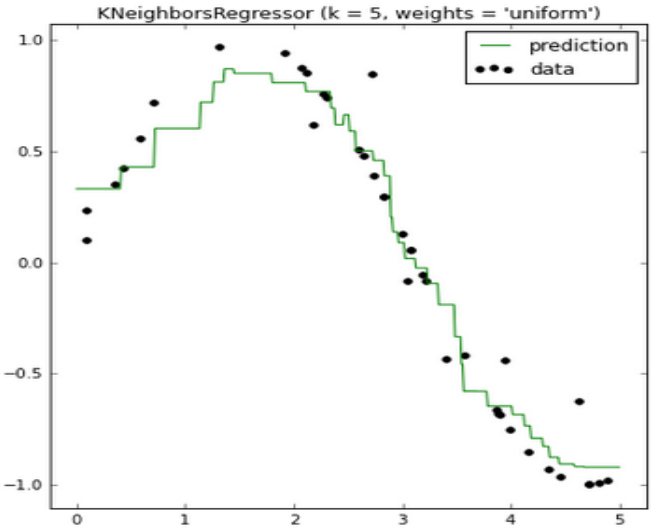
k-nearest neighbors algorithm

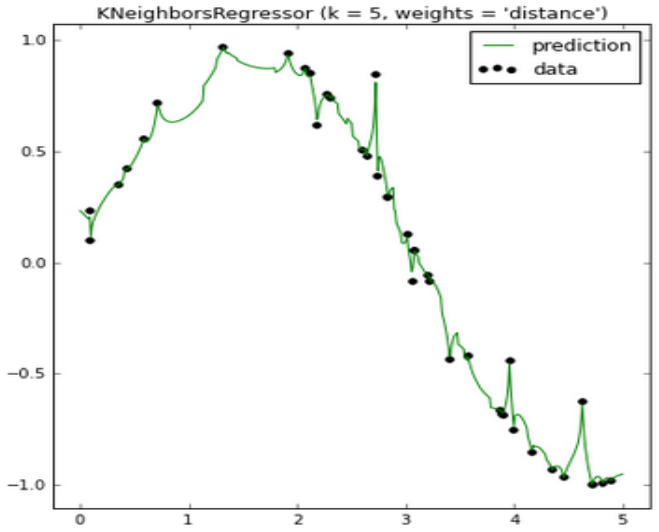
KNN regression :

Neighbor-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.

there are two possibilities two assign weights to neighbors, uniform and distance-based.

- in uniform: each point in the local neighborhood contributes uniformly to the classification of a query point
- in distance-based: nearby points contribute more to the regression than faraway points





Bayesian Nonparametric

why Bayesian nonparametric?

- Simplicity of the framework + complexity of the real world phenomena

Everything follows two simple rules:

Sum rule: $$P(x)=\sum_{y} P(x,y)$$

Product rule: $$P(x,y)=P(x)\,P(y|x)$$

Maximizing likelihood of observations:

$$P(\theta|D,m) = \frac{P(D|\theta,m)P(\theta|m)}{P(D|m)}$$

Inference: average over prediction of parameters

$$P(x|D,m) = \int P(x|D,\theta,m)\;P(\theta|D,m)\;d\theta$$

Bayesian Nonparametric

Real world application are sometimes complicated :

e.g.

- people's movie preference:
  - how many clusters?
  - even if few well defined clusters exist, how to estimate the distribution? is it Gaussian?!

- Natural Language Processing tasks like words segmentation, language modeling, grammar induction, etc.

Conclusion : any small finite number of parameters or limited models seems unreasonable to accomplish these task.

Gaussian Process

Gaussian Basics:

sampling from multivariate gaussian:

first consider sampling from a gaussian with parameters μ and σ^2.
we know, instead of sampling from this distribution, we can sample from a standard gaussian with mean 0 and variance 1, and then transform the results:

$$X_i \sim N(\mu, \sigma^2) \;=\; \mu + \sigma N(0,1)$$

same trick can be used to sample from multivariate gaussian:

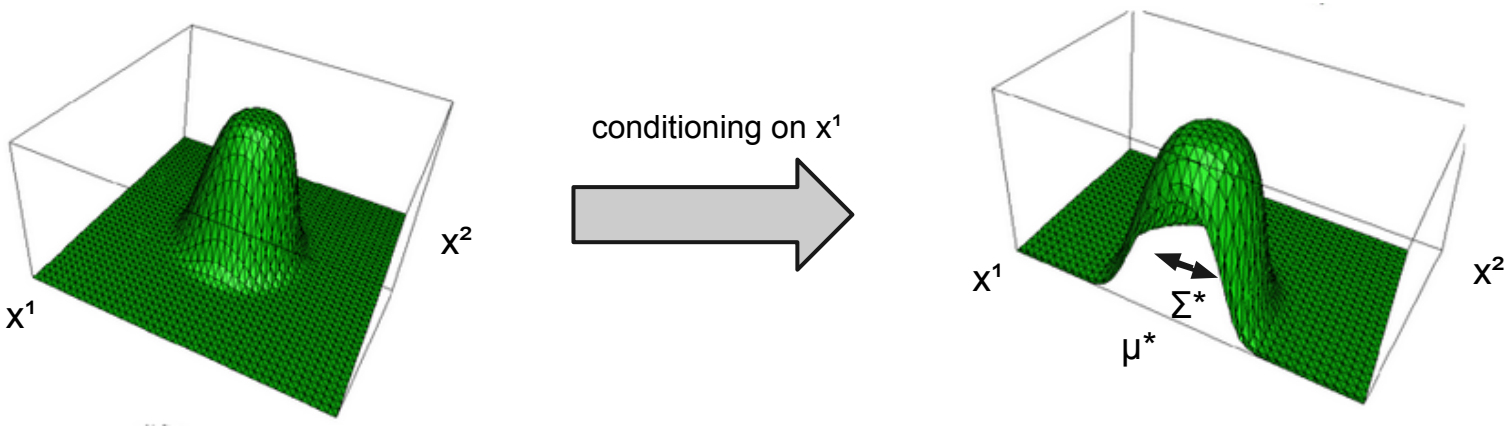
$$X_i \sim N\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \;=\; \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix} + LN\left(\begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)$$

where, L is the **Cholesky decomposition** of the covariance matrix Σ

Gaussian Process

Gaussian Basics:

conditioning on multivariate gaussian:



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

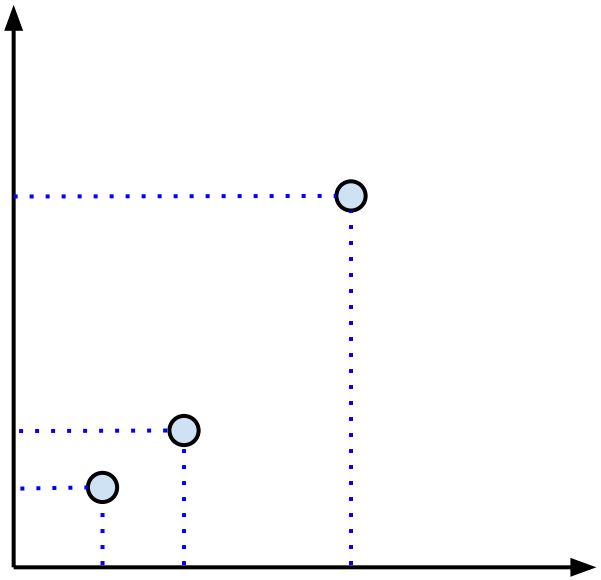
$$\mu^{2|1} = \mu^* = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$$

$$\Sigma^{2|1} = \Sigma^* = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

[http://paulbourke.net/geometry/polygonmesh/]

Gaussian Process

Basics:



assume points are distributed with gaussians like:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{21} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \right)$$

we expect points that are closer to be more correlated.
e.g.:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.8 & 0.2 \\ 0.8 & 1 & 0.5 \\ 0.2 & 0.5 & 1 \end{bmatrix} \right)$$

how to construct this matrix to measure the similarity of points?

one possible way is squared exponential:
other possibilities: periodic functions, etc.

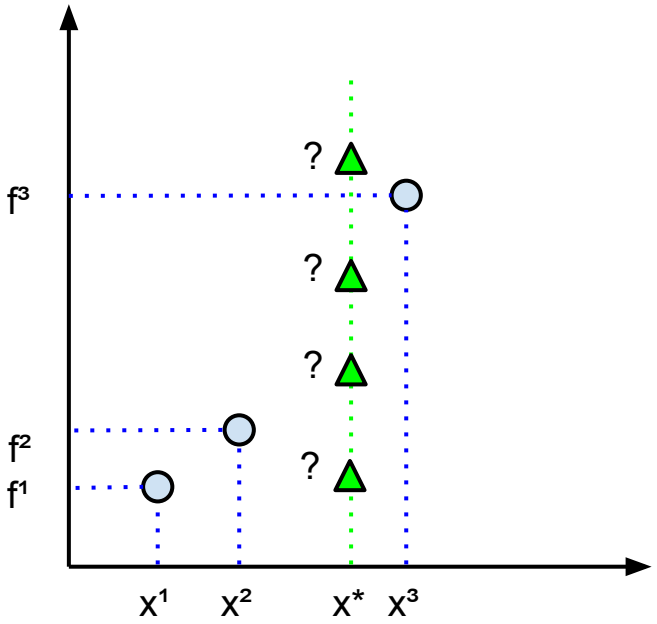
$$k_{ij} = e^{-\lambda \| x_i - x_j \|} = \begin{cases} 0 & \| x_i - x_j \| \longrightarrow \infty \\ 1 & x_i = x_j \end{cases}$$

reminder: covariance matrix is a measure of how strongly correlated a set of variables is

[Nando de Freitas]

Gaussian Process

now, imagine we get a new point x*, how to find the corresponding f*?



$$D=\{(x^1,f^1),(x^2,f^2),(x^3,f^3)\} \;,\; x^* \rightarrow f^*=?$$

since the point x* is closer to x³, we expect f* to also be closer to f³. this property called smoothness. (small variation in x results in small variation in f)

lets assume the point comes from the same distribution as other points:(i.e. a gaussian)

$$\underline{f} \sim N(0,K) \rightarrow f^* \sim N(0,k(x^*,x^*))$$

we want the new point to be correlated with others, so, we change the covariance matrix (i.e. kernel) as follows:

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} & \begin{bmatrix} k_{1*} \\ k_{2*} \\ k_{3*} \end{bmatrix} \\ \begin{bmatrix} k_{*1} & k_{*2} & k_{*3} \end{bmatrix} & k_{**} \end{bmatrix}\right)$$

[Nando de Freitas]

Gaussian Process

- A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic process governs the properties of functions.Gaussian process is defined as a distribution over functions

$$f \sim GP(m,k)$$

m = mean function, k=covariance function

one can loosely think of a function as a very long vector, each entry in the vector specifying the function value f (x) at a particular input x.

$$\boldsymbol{f} = (f(x_1), f(x_2), \ldots, f(x_n))$$

in Bayesian approach, we give a prior probability to every possible function where higher probabilities are given to functions that we consider to be more likely.

[C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006,]

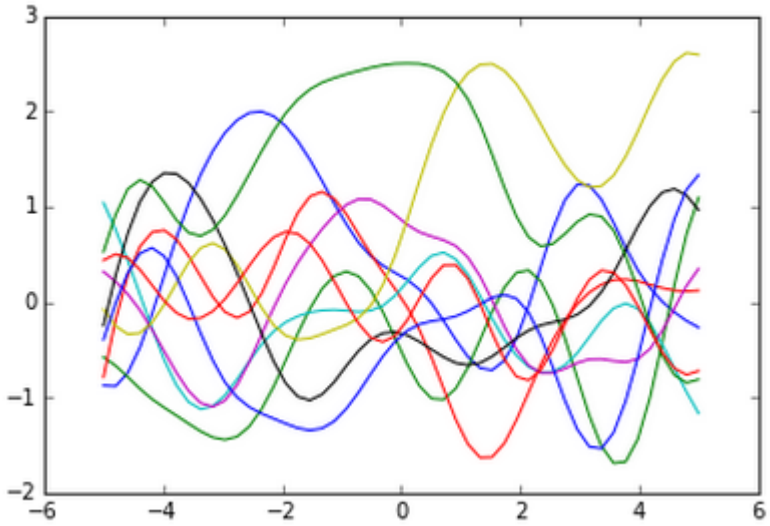
Gaussian Process

remember sampling from multivariate gaussian:

```
from __future__ import division
import numpy as np
import matplotlib.pyplot as pl

def kernel(a, b):
    """ GP squared exponential kernel """
    sqdist = np.sum(a**2,1).reshape(-1,1) + np.sum(b**2,1) - 2*np.dot(a, b.T)
    return np.exp(-.5 * sqdist)

n = 50 # number of test points.
Xtest = np.linspace(-5, 5, n).reshape(-1,1) # Test points.
K_ = kernel(Xtest, Xtest) # Kernel at test points.
# draw samples from the prior at our test points.
L = np.linalg.cholesky(K_ + 1e-6*np.eye(n))
f_prior = np.dot(L, np.random.normal(size=(n,10)))
pl.plot(Xtest, f_prior)
```



[Nando de Freitas]

Gaussian Process

remember conditioning in multivariate Gaussian:

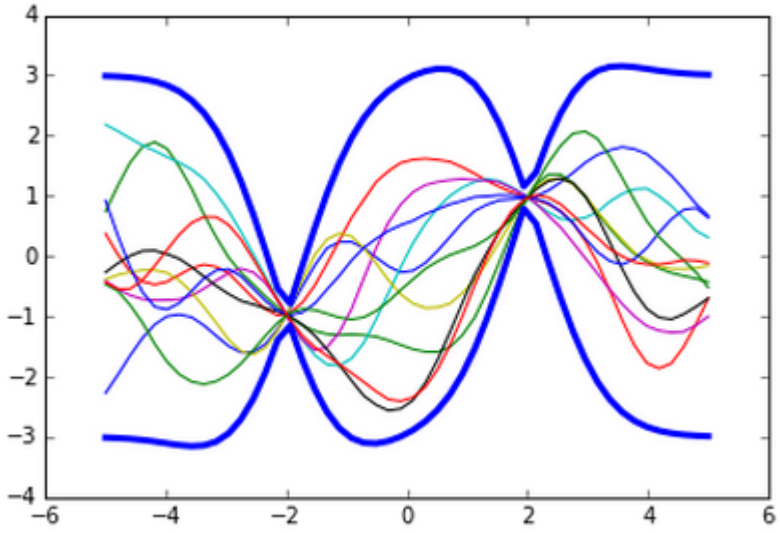
$$\mu^{2|1} = \mu^* = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$$
$$\Sigma^{2|1} = \Sigma^* = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

1 $\mathbf{L} = \text{cholesky}(\mathbf{K} + \sigma_y^2\mathbf{I})$;
2 $\boldsymbol{\alpha} = \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{y})$;
3 $\mathbb{E}[f_*] = \mathbf{k}_*^T\boldsymbol{\alpha}$;
4 $\mathbf{v} = \mathbf{L} \setminus \mathbf{k}_*$;
5 $\text{var}[f_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T\mathbf{v}$;
6 $\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T\boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{N}{2}\log(2\pi)$

numerical tricks:

$$\overline{f_*} = \mathbf{k}_*^T\mathbf{K}_y^{-1}\mathbf{y}$$

$$\mathbf{K}_y = \mathbf{L}\mathbf{L}^T$$
$$\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{y} = \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{y}$$

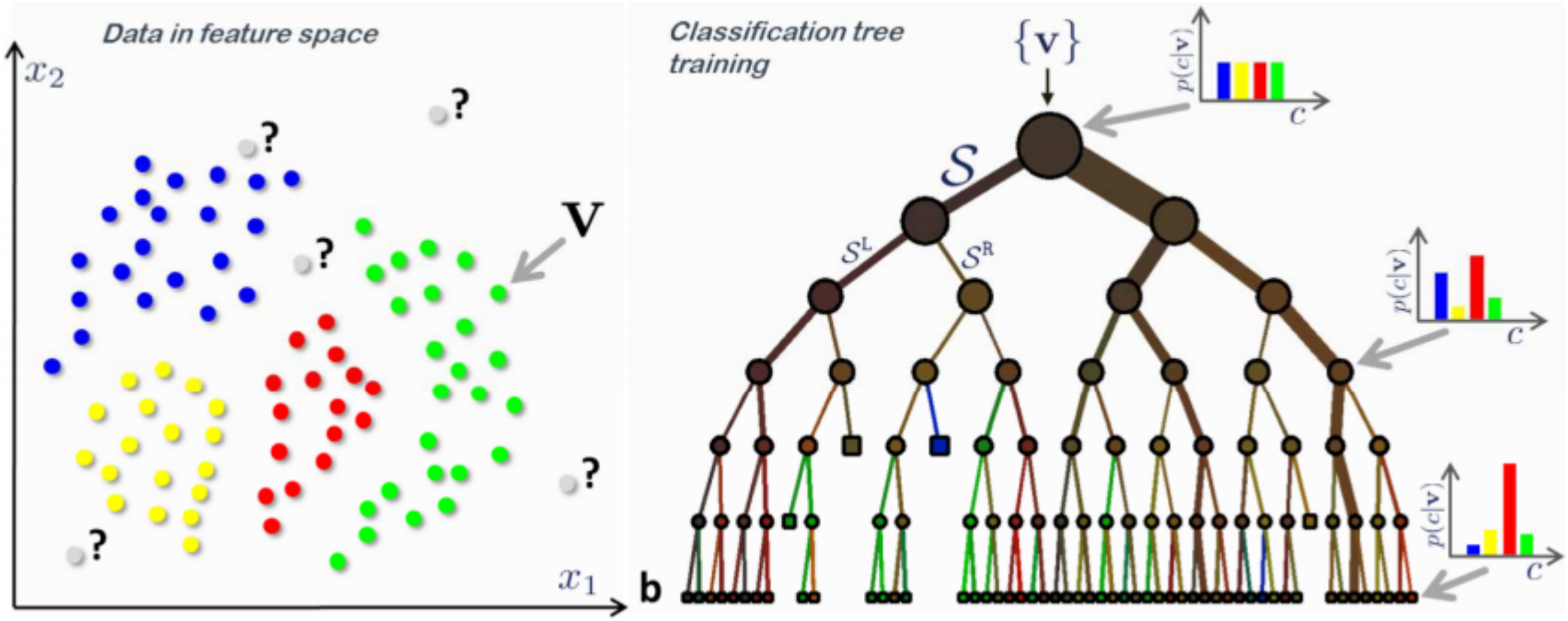


reminder: Cholesky decomposition or Cholesky factorization is a decomposition of a positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose

[Nando de Freitas]

Introduction to Decision Trees

Classification and Regression Trees or CART models, also called **decision trees**, are defined by recursively partitioning the input space, and defining a local model in each resulting region of the input space. this can be represented by a tree, by one leaf per region.



[Criminisi et al, 2011]

Introduction to Decision Trees

- each node represented by a histogram,.
- tree is binning the histogram using the training data
- we can compute the probability of each class for a test point, in any node of the tree, given the training data

$$Pr(c_i|Data,\ S) \propto h_{c_i}|S$$

- tree can be trained with any kind of data(attrib.). e.g. discrete, continuous

how to define decision in each node?
e.g. predict if a person will play tennis.

- Divide and Conquer:
  - Split the data into subsets
  - are they pure?
  - if yes: stop
  - if no: repeat

- See which subset new data falls into

Training examples: **9 yes / 5 no**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

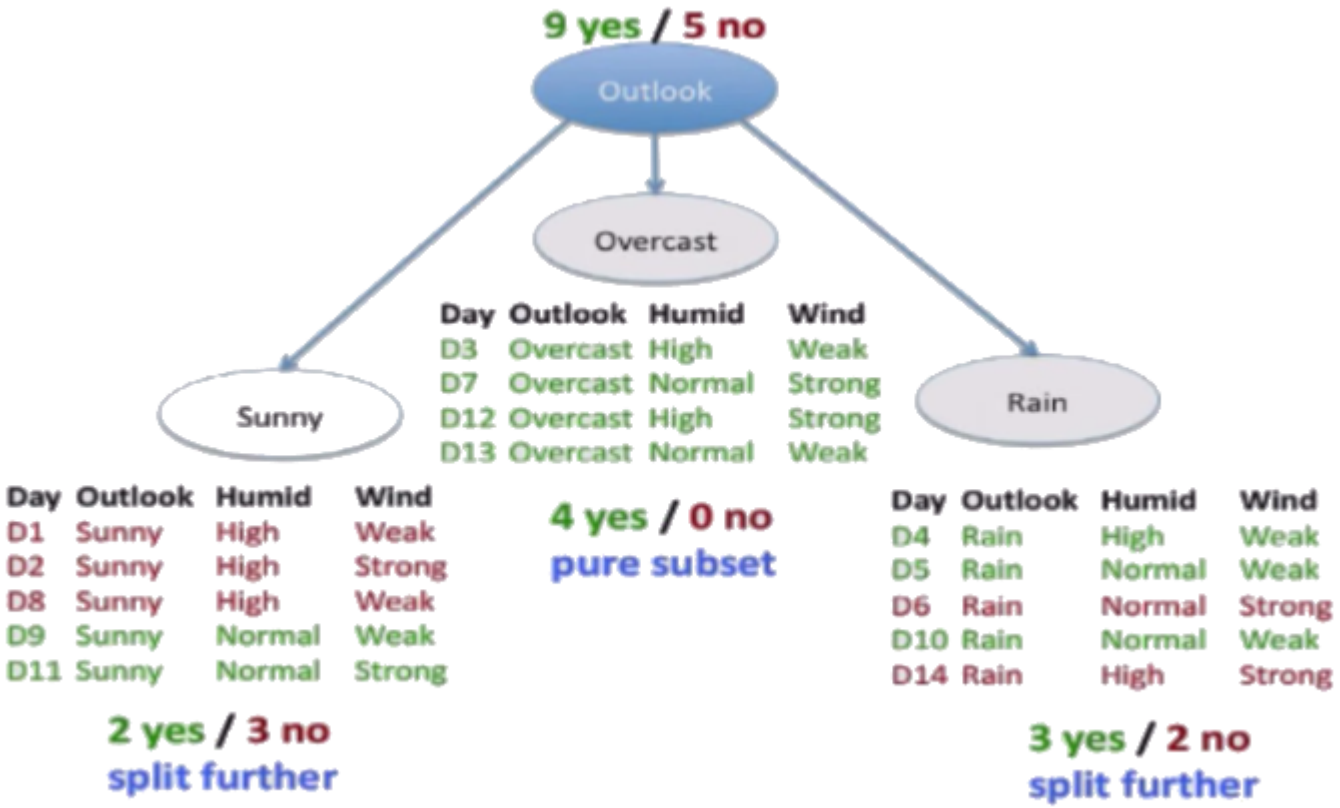
New data:

D15	Rain	High	Weak	?

[Victor Lavrenko]

Introduction to Decision Trees

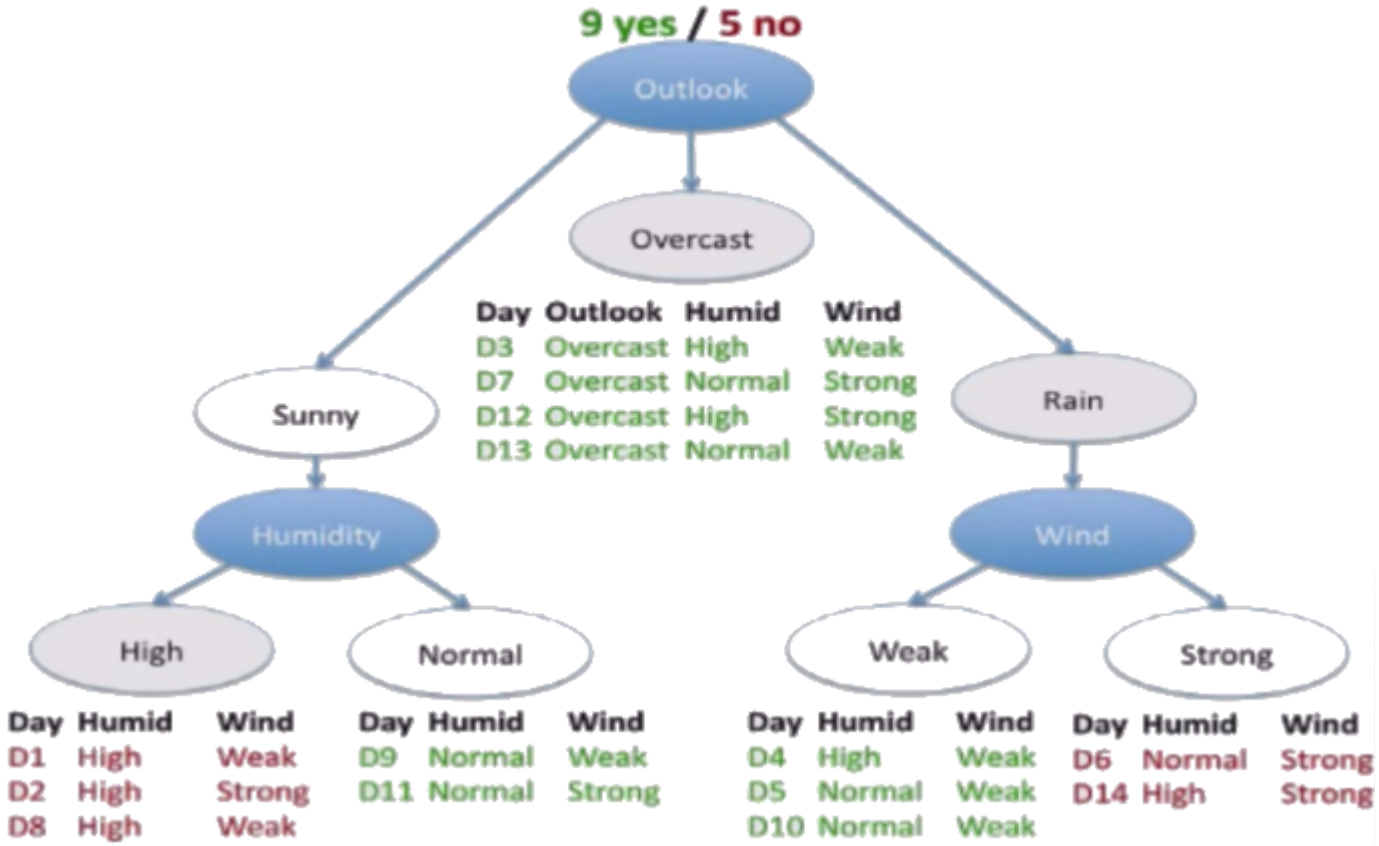
lets use “outlook” attribute as a decision:



[Victor Lavrenko]

Introduction to Decision Trees

now, in each daughter node, choose an attrib. as a decision that helps to split the data into pure classes:



[Victor Lavrenko]

Introduction to Decision Trees

in the end, use the resulting tree to make prediction on new data.

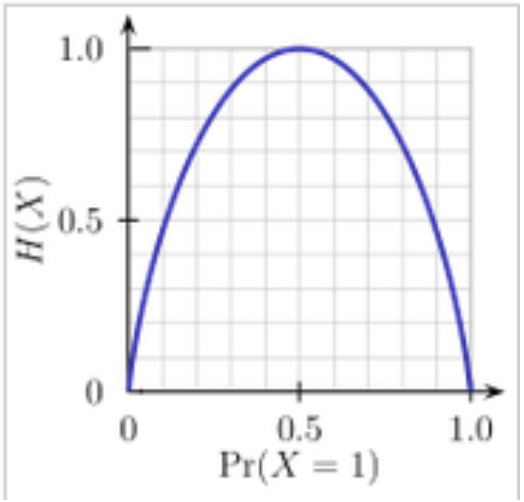
[Victor Lavrenko]

Introduction to Decision Trees

Reminder: Entropy

in information theory, entropy H is a measure of the uncertainty associated with a random variable.
it is defined as:

$$H(X) = -\sum_{x} P(x|\theta) \log P(x|\theta)$$



for Bernoulli distribution, where probability is defined as:

$$P(x|\theta) = \theta^{x}(1-\theta)^{1-x} = \begin{cases} \theta & if \quad x=1 \\ 1-\theta & if \quad x=0 \end{cases}$$

entropy is:

$$H(X) = -\sum_{x} \theta^{x}(1-\theta)^{1-x} \log(\theta^{x}(1-\theta)^{1-x}) = -(\theta \log(\theta) + (1-\theta)\log(1-\theta))$$

for a fair coin, entropy (uncertainty) is maximum, where probability of H or T are equal.

Introduction to Decision Trees

- entropy can be generalized for discrete random variable X with N possible outcomes:

$$H(X)= \sum_{i=1}^{N} P(x_i)log(P(x_i))$$

reminder: maximum likelihood is just a way to minimize our uncertainty about the world.

now, we can leverage the concept of entropy to introduce the notion of **Information Gain** in nodes of a tree as follows:

$$I_j = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} H(S_j^i) \qquad \Longrightarrow Ij \uparrow \propto \sum H^i \downarrow$$

Here information gain, is defined as the difference between the entropy of a node and its leaves. In other words, it is a measure of reduction in our uncertainty about the data by splitting it into leaves using the decision threshold that we chose at that node.
In the above formula, we are summing the weighted entropy of the leaves (here we used a binary decision), where weights are proportion of the data that ended up at each leaf using our decision criteria.

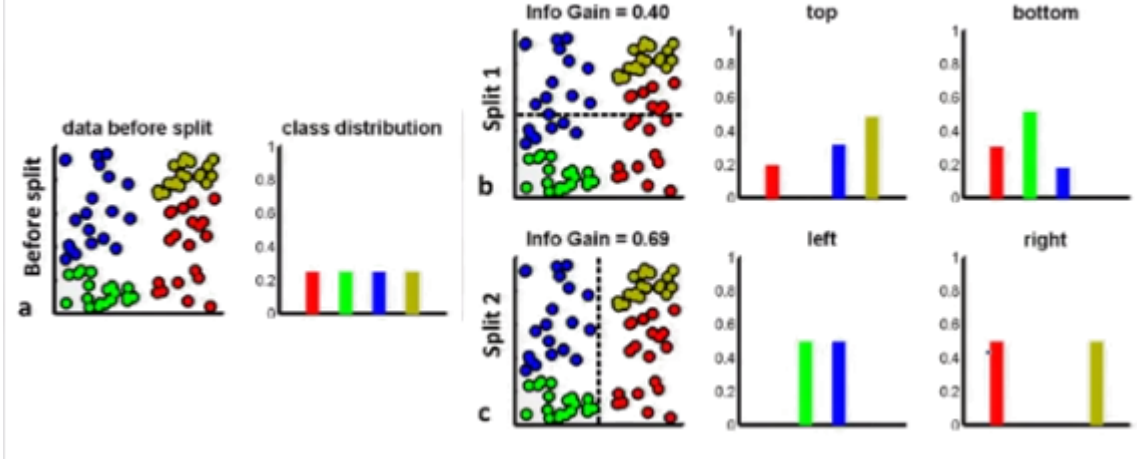
[Criminisi et al, 2011]

Introduction to Decision Trees

example: splitting data in a 2D plane using lines.

which line gives us more information about the data?

$$I_j = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} H(S_j^i)$$



lets do the calculation: assume the highest value in daughter histogram is 3 and the lowest is 1. and sum of point that are ended up in each leaf is 6. (using log2)

for split 1:

$$2*(\frac{1}{2}*-1*(\frac{1}{6}\log\frac{1}{6}+\frac{2}{6}\log\frac{2}{6}+\frac{3}{6}\log\frac{3}{6}))=1.459$$

for split 2:

$$2*(\frac{1}{2}*-1*(\frac{3}{6}\log\frac{3}{6}+\frac{3}{6}\log\frac{3}{6}))=1.$$

$$\Longrightarrow H(split\_1) < H(split\_2)$$

entropy of parent node:

$$-1*4*(\frac{3}{12}\log\frac{3}{12})=2.$$

[Criminisi et al, 2011]

Random Forest

Here is the problem, assume you have millions of data with order of 100,000 features, is it possible to calculate the expected information gain of all the features for each node?

solution: **Random Forest**

algorithm:

- for b=1 to B (number of trees):
  - Draw a bootstrap sample **Z*** of size N from the training data
  - Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
    - i. select *m* variables at random from the *p* variables.
    - ii. pick the best variable/split-point among the *m.*
    - iii. split the node into two daughter nodes.
- output the ensemble of trees. $\{T_b\}_1^B$

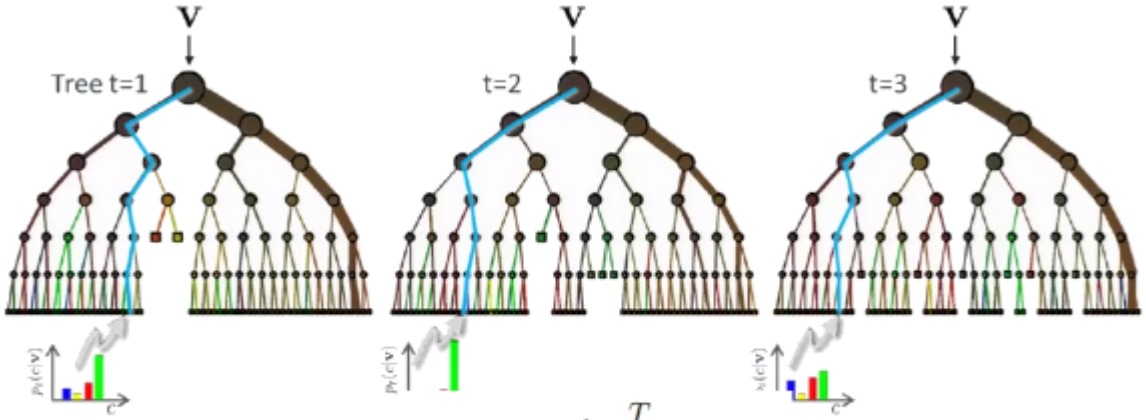
reminder:
bootstrap: random choosing subset of data, with substitution
bagging: construct models based on different subsets of data

[From the book of Hastie, Friedman and Tibshirani]

Random Forest

notice:

- Each tree is made of only a subset of data, with considering only part of the feature vector, so individual trees are high variance models. They only capture small details in subset of data.
- To fix this problem, we make lots of trees and combine the results from all those trees, to make prediction. when the number of trees grow, the solution converges to optimal case.
- All trees are trained **independently** (and possibly in **parallel**).
- Each test point is simultaneously pushed through all trees (starting from the root) until it reaches the corresponding leaves.



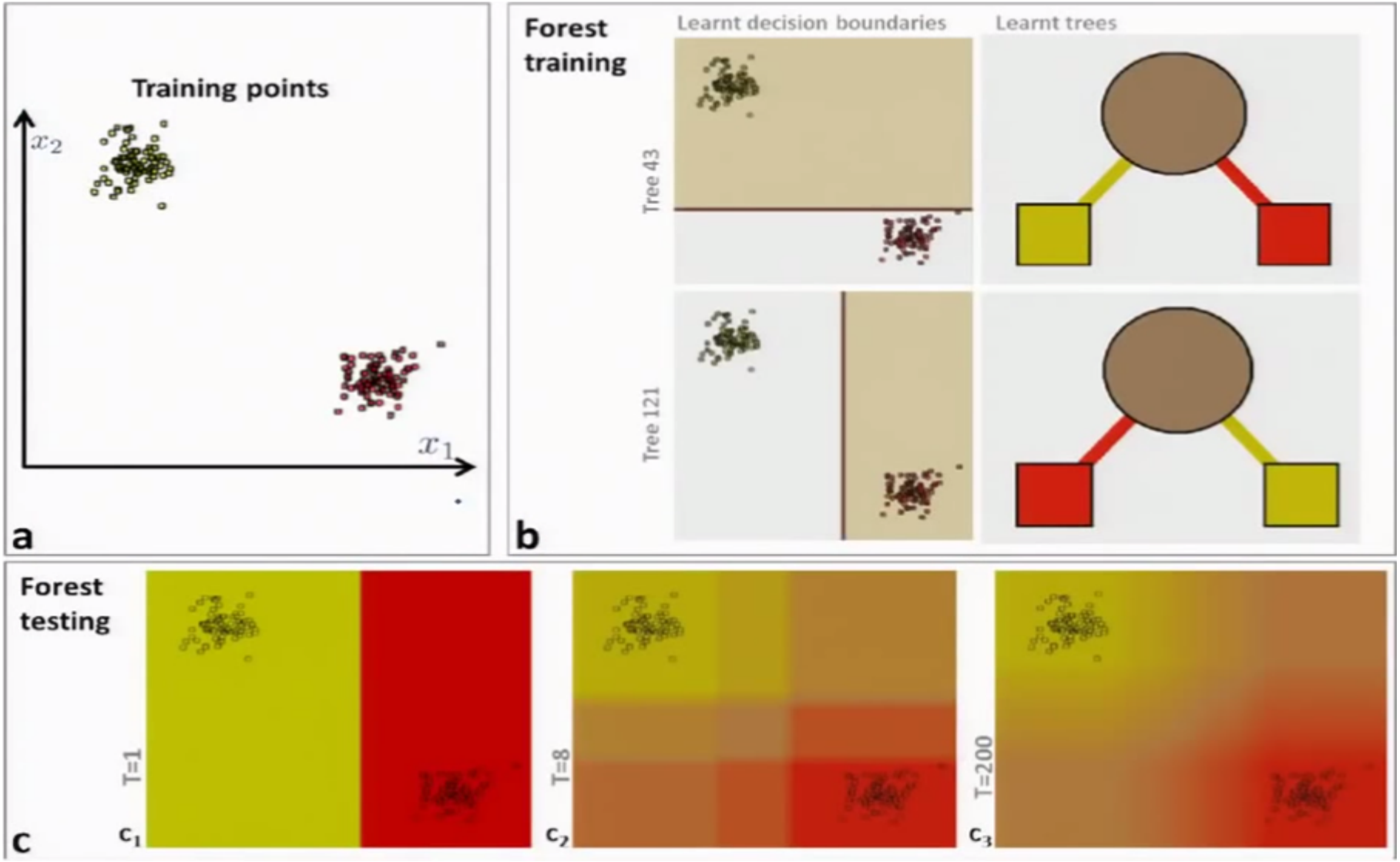
$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$$

[Criminisi et al, 2011]

Random Forest

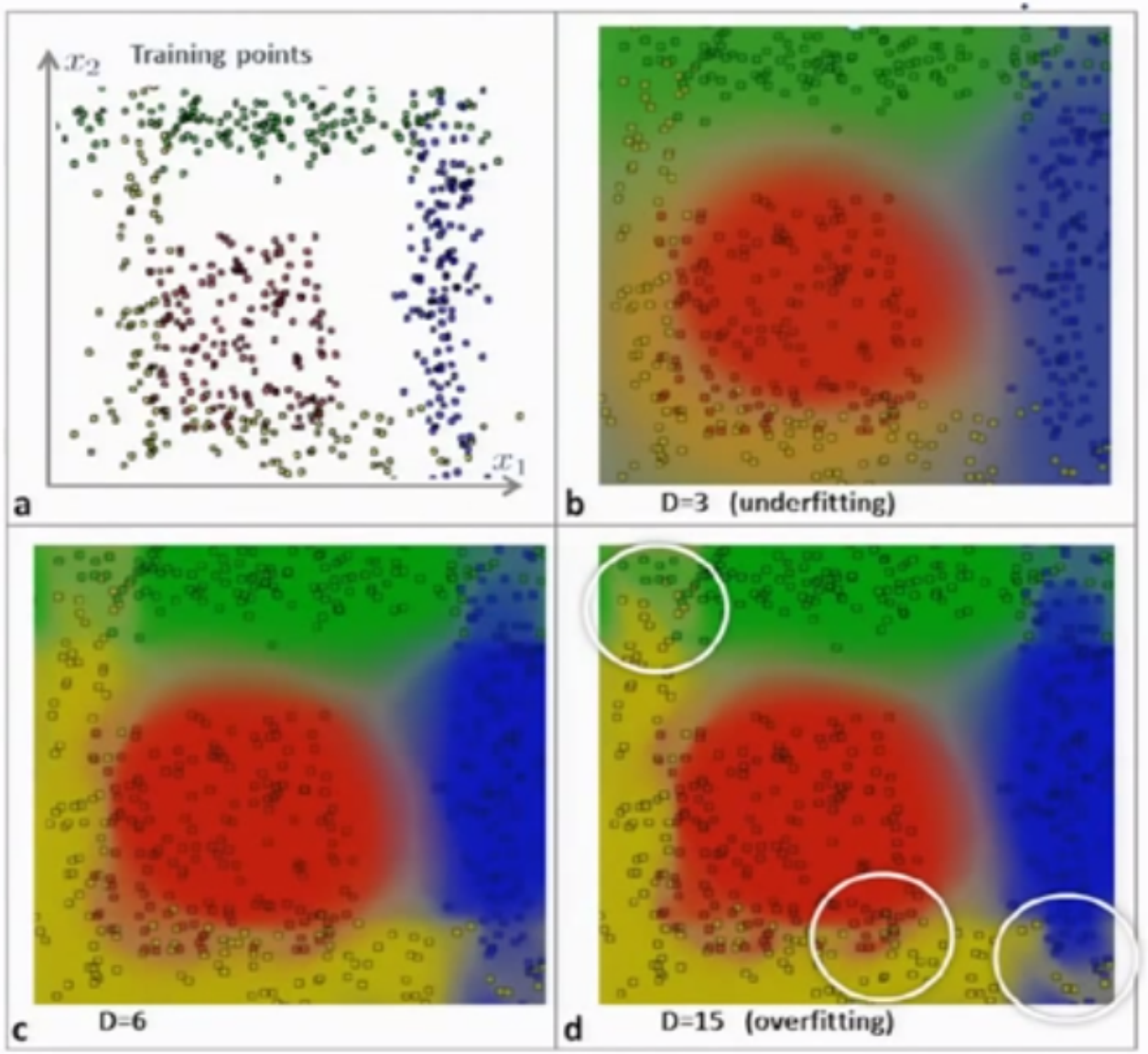
Effect of Forest size:

[Criminisi et al, 2011]



Random Forest

Effect of depth:

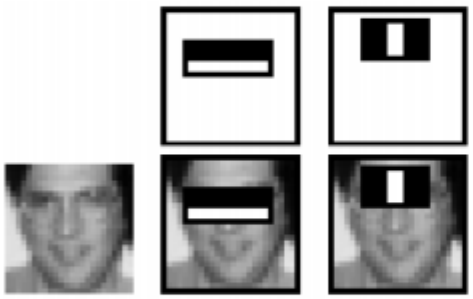


[Criminisi et al, 2011]

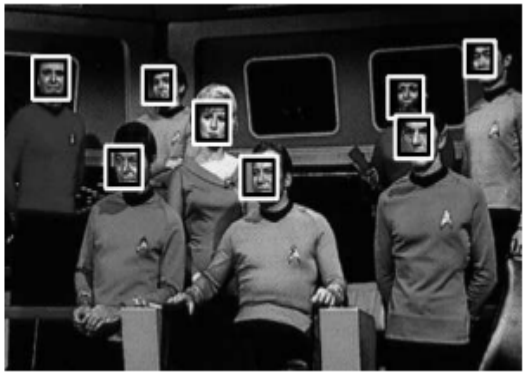
Random Forest

Application: Object Detection

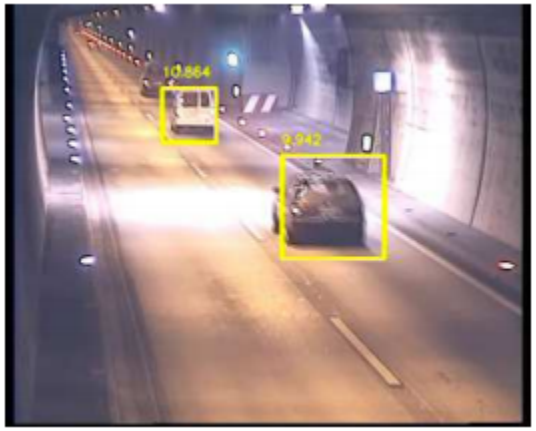
Idea: Extract simple features from all 24 by 24 pixel patches x_i. E.g., the value of a *two-rectangle feature is the difference between the sum* of the pixels within two rectangular regions. Then compare the level of activation (value of the feature f) with respect to a threshold (theta).

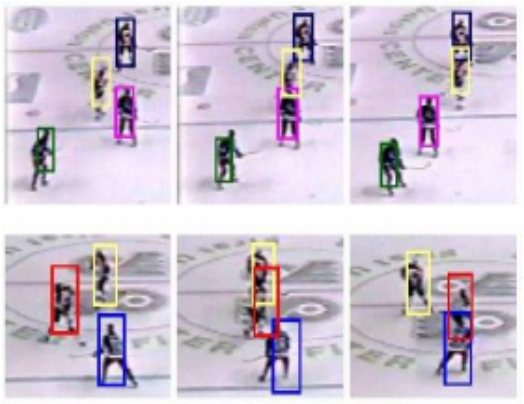


$$h_t(x_i) = \begin{cases} 1 & \text{if } f_t(x_i) > \theta_t \\ 0 & \text{otherwise} \end{cases}$$



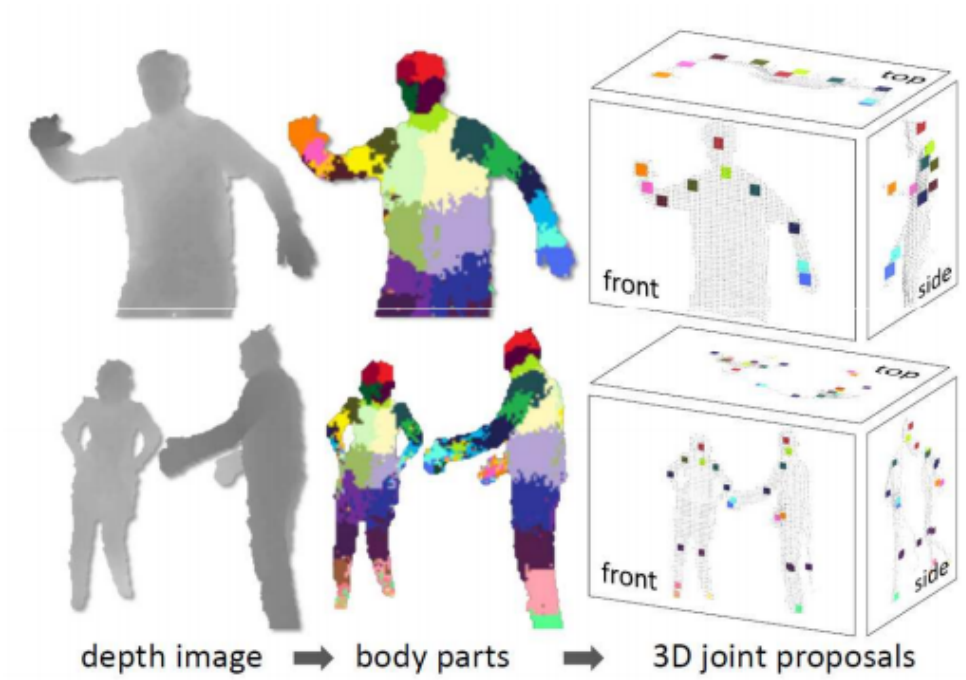
Relevant feature Irrelevant feature





Random Forest

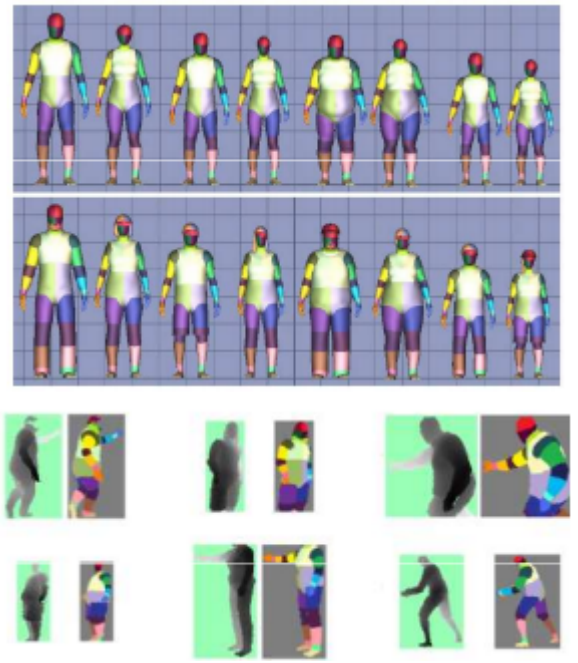
Random Forest and Kinect: Pose estimation

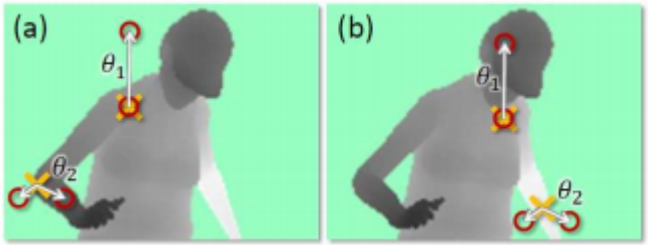


- use lots of simple depth features to construct random forest(e.g. relative depth of the points)

[Jamie Shotton et al 2011]

- using computer graphic to generate plenty of data





Conclusion

- Modern data acquisition routinely produces massive and complex datasets,
  - image data from functional Magnetic Resonance Imaging (fMRI)
  - climate data from geographically distributed data centers.
  - etc.
- Existing high dimensional theories and learning algorithms rely heavily on parametric models, which assume the data come from an underlying distribution (e.g. Gaussian or linear models) that can be characterized by a finite number of parameters.
- Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions
- Non-parametric methods allow the statistical modeller to be flexible with the dimensions and structure of their model