

INTRODUCTION TO STATISTICAL MACHINE LEARNING

Marcus Hutter

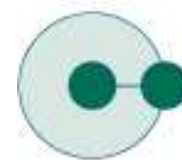
Canberra, ACT, 0200, Australia



ANU



RSISE



NICTA

Abstract

This course provides a broad introduction to the methods and practice of statistical machine learning, which is concerned with the development of algorithms and techniques that learn from observed data by constructing stochastic models that can be used for making predictions and decisions. Topics covered include Bayesian inference and maximum likelihood modeling; regression, classification, density estimation, clustering, principal component analysis; parametric, semi-parametric, and non-parametric models; basis functions, neural networks, kernel methods, and graphical models; deterministic and stochastic optimization; overfitting, regularization, and validation.

Table of Contents

1. Introduction / Overview / Preliminaries
2. Linear Methods for Regression
3. Nonlinear Methods for Regression
4. Model Assessment & Selection
5. Large Problems
6. Unsupervised Learning
7. Sequential & (Re)Active Settings
8. Summary

1 INTRO/OVERVIEW/PRELIMINARIES

- What is Machine Learning? Why Learn?
- Related Fields
- Applications of Machine Learning
- Supervised \leftrightarrow Unsupervised \leftrightarrow Reinforcement Learning
- Dichotomies in Machine Learning
- Mini-Introduction to Probabilities

What is Machine Learning?

Machine Learning is concerned with the development of algorithms and techniques that allow computers to **learn**

Learning in this context is the process of gaining understanding by constructing models of observed data with the intention to use them for prediction.

Related fields

- **Artificial Intelligence:** smart algorithms
- **Statistics:** inference from a sample
- **Data Mining:** searching through large volumes of data
- **Computer Science:** efficient algorithms and complex models

Why 'Learn' ?

There is no need to “learn” to calculate payroll

Learning is used when:

- Human expertise does not exist (navigating on Mars),
- Humans are unable to explain their expertise (speech recognition)
- Solution changes in time (routing on a computer network)
- Solution needs to be adapted to particular cases (user biometrics)

Example: It is easier to write a program that *learns* to play checkers or backgammon well by self-play rather than converting the expertise of a master player to a program.

Handwritten Character Recognition

an example of a difficult machine learning problem

7	2	1	0	4	1	4	9	5	9
0	6	9	0	1	5	9	7	8	4
9	6	6	5	4	0	7	4	0	1
3	1	3	4	7	2	7	1	2	1
1	7	4	2	3	5	1	2	4	4
6	3	5	5	6	0	4	1	9	5
7	8	9	3	7	4	6	4	3	0
7	0	2	9	1	7	3	2	9	7
7	6	2	7	8	4	7	3	6	1
3	6	9	3	1	4	1	7	6	9

Task: Learn general mapping from pixel images to digits from examples

Applications of Machine Learning

machine learning has a wide spectrum of applications including:

- natural language processing,
- search engines,
- medical diagnosis,
- detecting credit card fraud,
- stock market analysis,
- bio-informatics, e.g. classifying DNA sequences,
- speech and handwriting recognition,
- object recognition in computer vision,
- playing games – learning by self-play: Checkers, Backgammon.
- robot locomotion.

Some Fundamental Types of Learning

- Supervised Learning
 - Classification
 - Regression
- Unsupervised Learning
 - Association
 - Clustering
 - Density Estimation
- Reinforcement Learning
 - Agents
- Others
 - SemiSupervised Learning
 - Active Learning

Supervised Learning

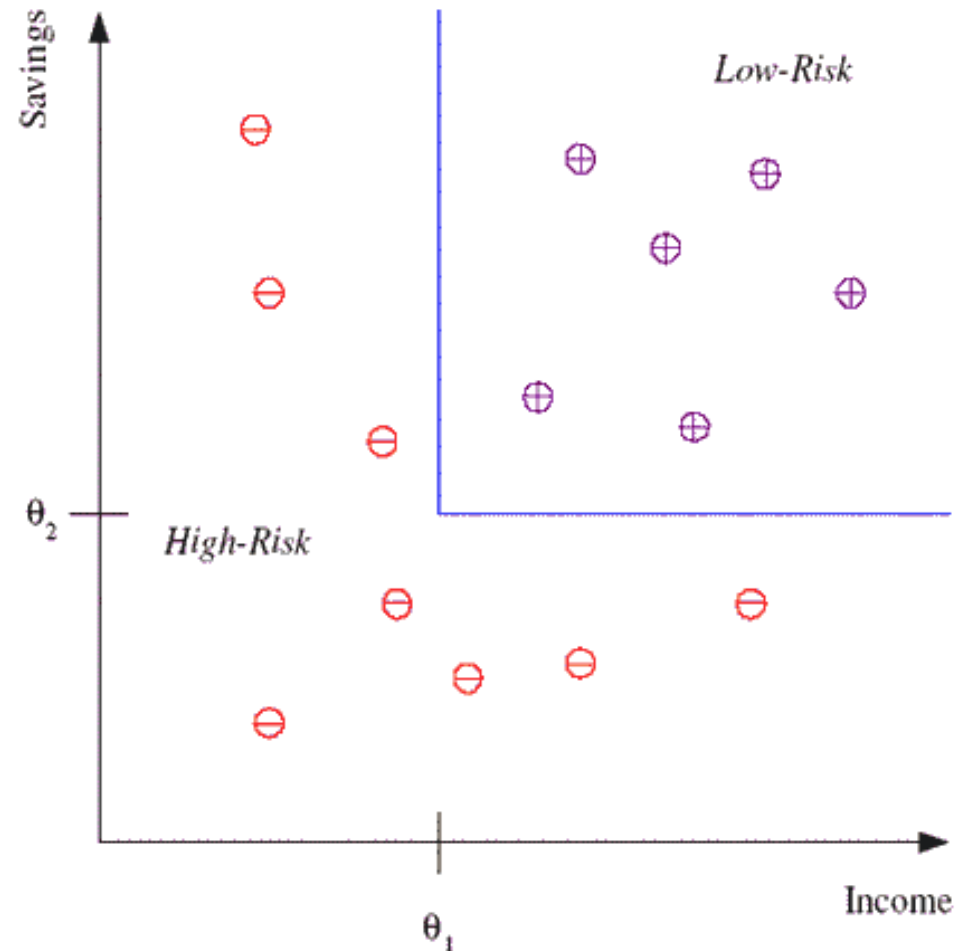
- Prediction of future cases:
Use the rule to predict the output for future inputs
- Knowledge extraction:
The rule is easy to understand
- Compression:
The rule is simpler than the data it explains
- Outlier detection:
Exceptions that are not covered by the rule, e.g., fraud

Classification

Example:

Credit scoring

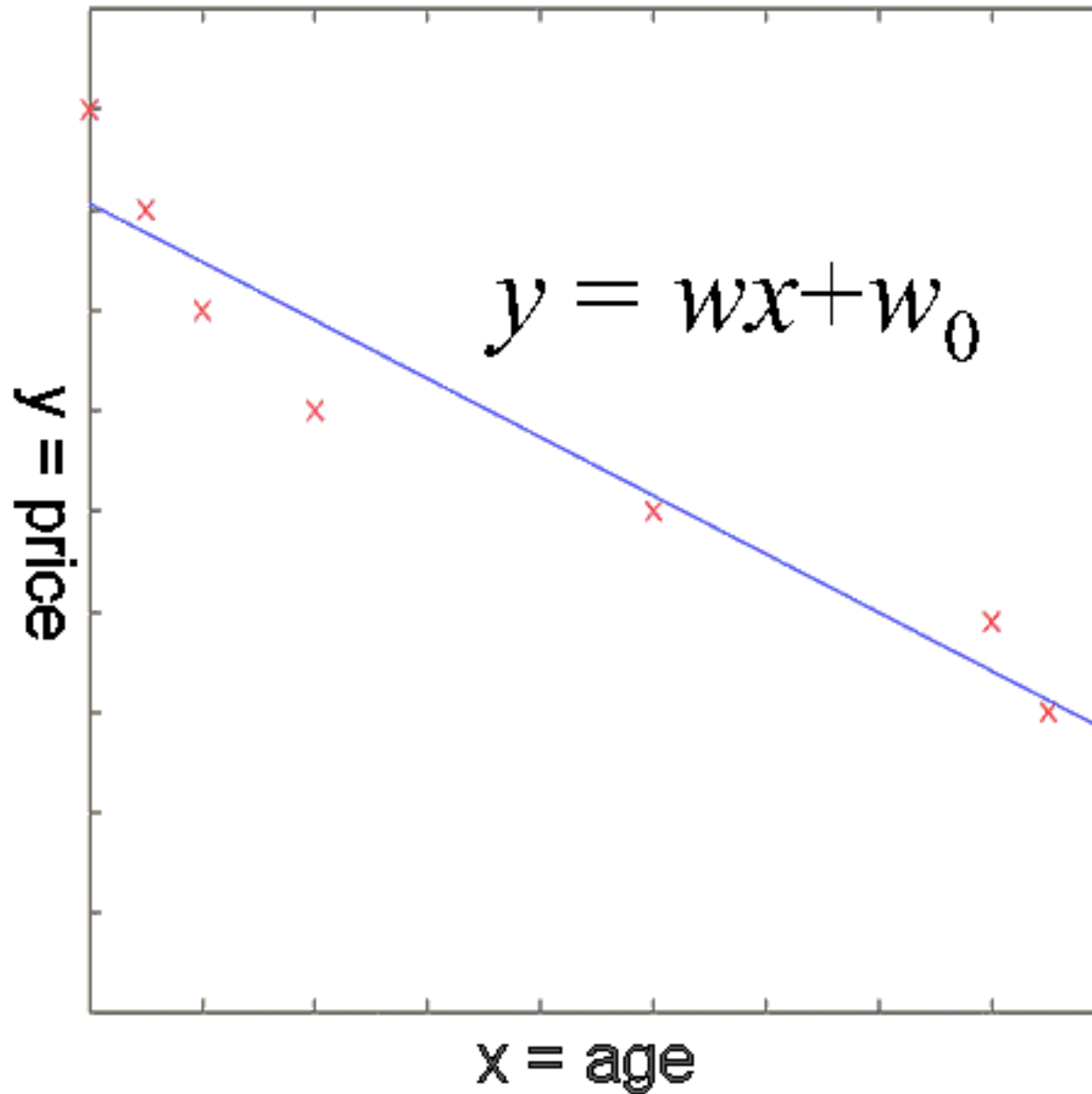
Differentiating between
low-risk and high-risk
customers from their
Income and *Savings*



Discriminant: IF $income > \theta_1$ AND $savings > \theta_2$
THEN low-risk ELSE high-risk

Regression

Example: Price $y = f(x) + \text{noise}$ of a used car as function of age x



Unsupervised Learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Example applications:
 - Customer segmentation in CRM
 - Image compression: Color quantization
 - Bioinformatics: Learning motifs

Reinforcement Learning

- Learning a policy: A sequence of outputs
- No supervised output but delayed reward
- Credit assignment problem
- Game playing
- Robot in a maze
- Multiple agents, partial observability, ...

Dichotomies in Machine Learning

scope of my lecture		⇔	scope of other lectures	
(machine) learning	⇔	(GOFAI) knowledge-based		
statistical	⇔	logic-based		
induction	⇔	prediction	⇔	decision ⇔ action
		regression	⇔	classification
independent identically distributed	⇔	non-iid		
online learning	⇔	offline/batch learning		
passive prediction	⇔	active learning		
parametric	⇔	non-parametric		
conceptual/mathematical	⇔	computational issues		
exact/principled	⇔	heuristic		
supervised learning	⇔	unsupervised	⇔	RL learning

Probability Basics

**Probability is used to describe uncertain events;
the chance or belief that something is or will be true.**

Example: Fair Six-Sided Die:

- **Sample space:** $\Omega = \{1, 2, 3, 4, 5, 6\}$
- **Events:** Even = $\{2, 4, 6\}$, Odd = $\{1, 3, 5\} \subseteq \Omega$
- **Probability:** $P(6) = \frac{1}{6}$, $P(\text{Even}) = P(\text{Odd}) = \frac{1}{2}$
- **Outcome:** $6 \in E$.
- **Conditional probability:** $P(6|\text{Even}) = \frac{P(6 \text{ and Even})}{P(\text{Even})} = \frac{1/6}{1/2} = \frac{1}{3}$

General Axioms:

- $P(\{\}) = 0 \leq P(A) \leq 1 = P(\Omega)$,
- $P(A \cup B) + P(A \cap B) = P(A) + P(B)$,
- $P(A \cap B) = P(A|B)P(B)$.

Probability Jargon

Example: (Un)fair coin: $\Omega = \{\text{Tail, Head}\} \simeq \{0, 1\}$. $P(1) = \theta \in [0, 1]$:

Likelihood: $P(1101|\theta) = \theta \times \theta \times (1 - \theta) \times \theta$

Maximum Likelihood (ML) estimate: $\hat{\theta} = \arg \max_{\theta} P(1101|\theta) = \frac{3}{4}$

Prior: If we are indifferent, then $P(\theta) = \text{const.}$

Evidence: $P(1101) = \sum_{\theta} P(1101|\theta)P(\theta) = \frac{1}{20}$ (actually \int)

Posterior: $P(\theta|1101) = \frac{P(1101|\theta)P(\theta)}{P(1101)} \propto \theta^3(1 - \theta)$ (**BAYES RULE!**).

Maximum a Posterior (MAP) estimate: $\hat{\theta} = \arg \max_{\theta} P(\theta|1101) = \frac{3}{4}$

Predictive distribution: $P(1|1101) = \frac{P(11011)}{P(1101)} = \frac{2}{3}$

Expectation: $\mathbf{E}[f|\dots] = \sum_{\theta} f(\theta)P(\theta|\dots)$, e.g. $\mathbf{E}[\theta|1101] = \frac{2}{3}$

Variance: $\text{Var}(\theta) = \mathbf{E}[(\theta - \mathbf{E}\theta)^2|1101] = \frac{2}{63}$

Probability density: $P(\theta) = \frac{1}{\varepsilon}P([\theta, \theta + \varepsilon])$ for $\varepsilon \rightarrow 0$

2 LINEAR METHODS FOR REGRESSION

- Linear Regression
- Coefficient Subset Selection
- Coefficient Shrinkage
- Linear Methods for Classification
- Linear Basis Function Regression (LBFR)
- Piecewise linear, Splines, Wavelets
- Local Smoothing & Kernel Regression
- Regularization & 1D Smoothing Splines

Linear Regression

fitting a linear function to the data

- Input “feature” vector $\mathbf{x} := (1 \equiv x^{(0)}, x^{(1)}, \dots, x^{(d)}) \in \mathbb{R}^{d+1}$
- Real-valued noisy response $y \in \mathbb{R}$.

- Linear regression model:

$$\hat{y} = f_{\mathbf{w}}(\mathbf{x}) = w_0 x^{(0)} + \dots + w_d x^{(d)}$$

- Data: $D = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$

- Error or loss function:

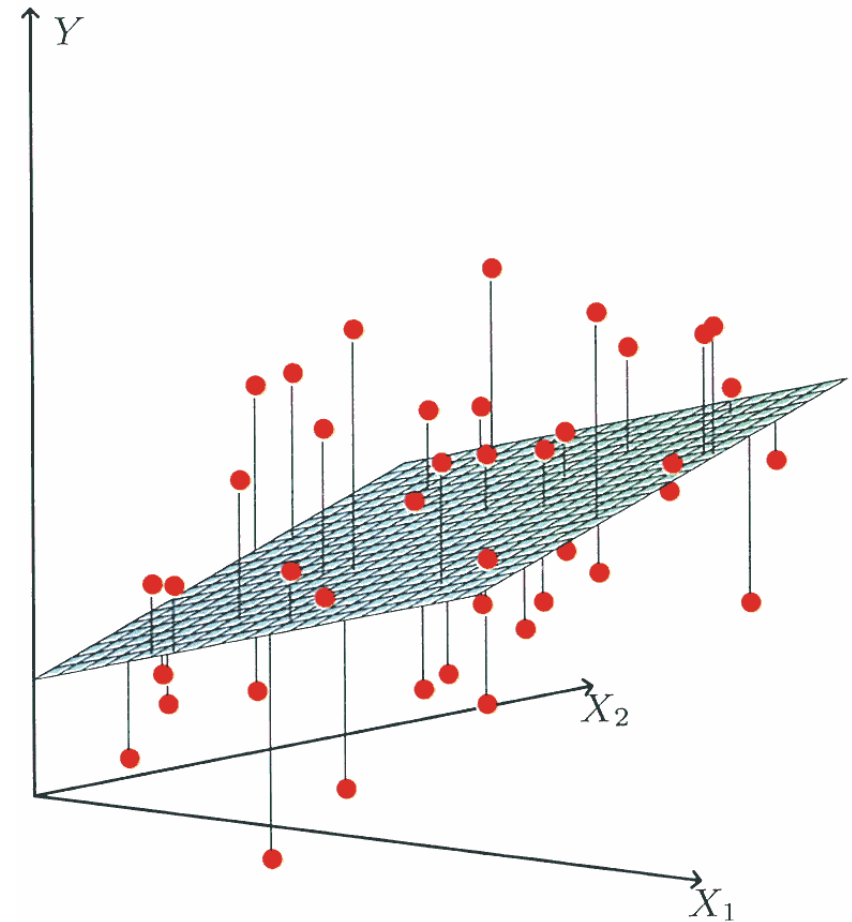
Example: Residual sum of squares:

$$\text{Loss}(\mathbf{w}) = \sum_{i=1}^n (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

- Least squares (LSQ) regression:

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \text{Loss}(\mathbf{w})$$

- Example: Person’s weight y as a function of age x^1 , height x^2 .



Coefficient Subset Selection

Problems with least squares regression if d is large:

- **Overfitting:** The plane fits the data well (perfect for $d \geq n$), but predicts (generalizes) badly.
- **Interpretation:** We want to identify a small subset of features important/relevant for predicting y .

Solution 1: Subset selection:

Take those k out of d features that minimize the LSQ error.

Coefficient Shrinkage

Solution 2: Shrinkage methods:

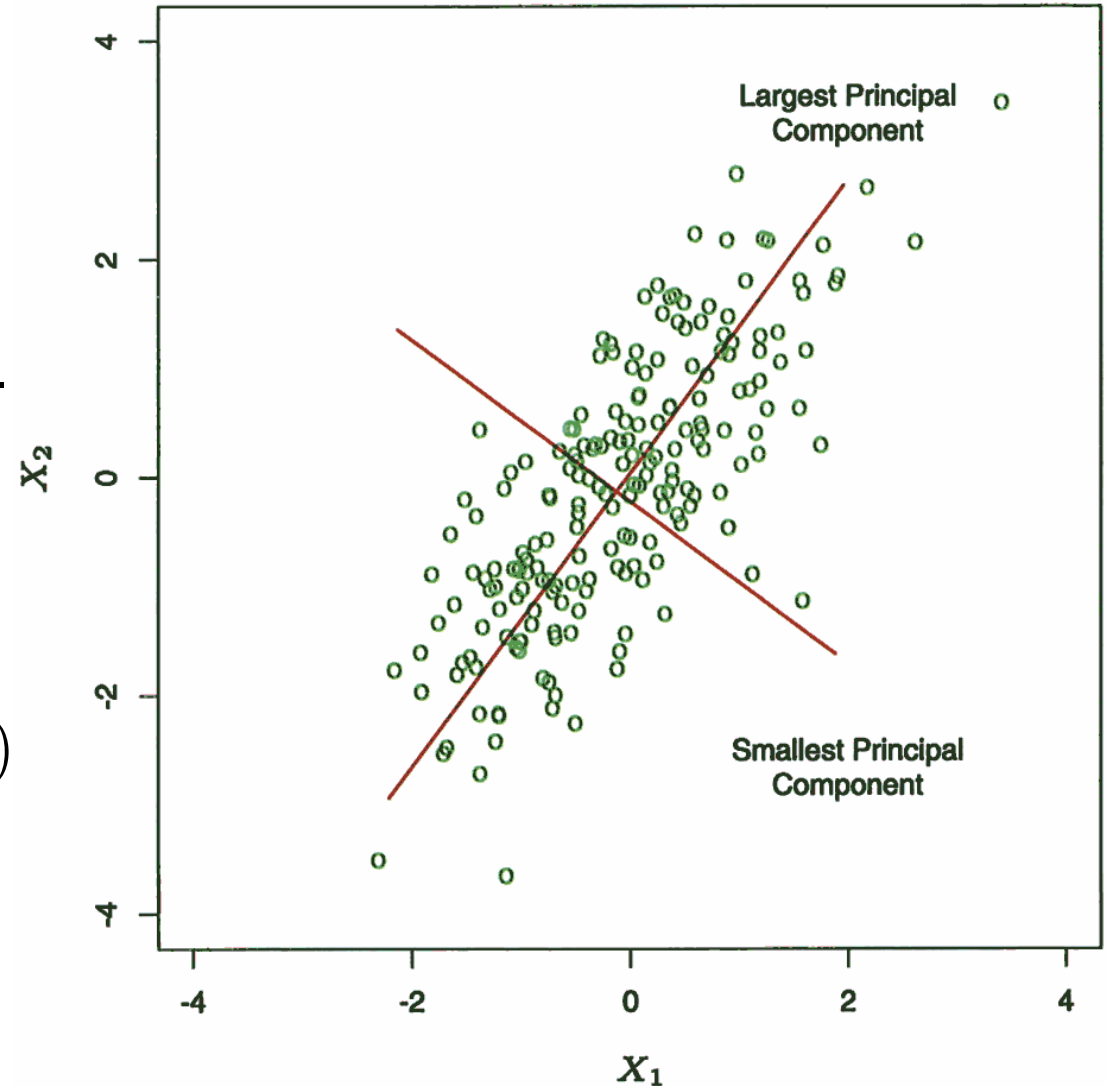
Shrink the least squares w
by penalizing the Loss:

Ridge regression: Add $\propto \|w\|_2^2$.

Lasso: Add $\propto \|w\|_1$.

Bayesian linear regression:

Comp. MAP $\arg \max_w P(w|D)$
from prior $P(w)$ and
sampling model $P(D|w)$.



Weights of low variance components shrink most.

Linear Methods for Classification

Example: $\mathcal{Y} = \{\text{spam, non-spam}\} \simeq \{-1, 1\}$ (or $\{0, 1\}$)

Reduction to regression: Regard $y \in \mathbb{R} \Rightarrow \hat{w}$ from linear regression.

Binary classification: If $f_{\hat{w}}(x) > 0$ then $\hat{y} = 1$ else $\hat{y} = -1$.

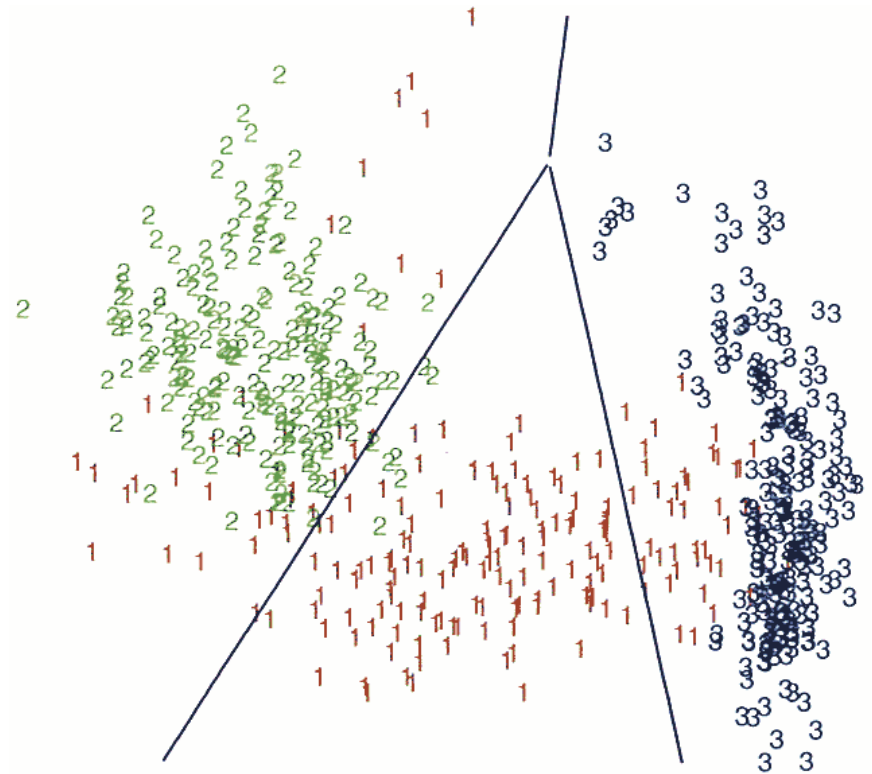
Probabilistic classification: Predict probability that new x is in class y .

$$\text{Log-odds } \log \frac{P(y=1|x, D)}{P(y=0|x, D)} := f_{\hat{w}}(x)$$

Improvements:

- Linear Discriminant Analysis (LDA)
- Logistic Regression
- Perceptron
- Maximum Margin Hyperplane
- Support Vector Machine

Generalization to non-binary \mathcal{Y} possible.



Linear Basis Function Regression (LBFR)

= powerful generalization of and reduction to linear regression!

Problem: Response $y \in \mathbb{R}$ is often not linear in $\mathbf{x} \in \mathbb{R}^d$.

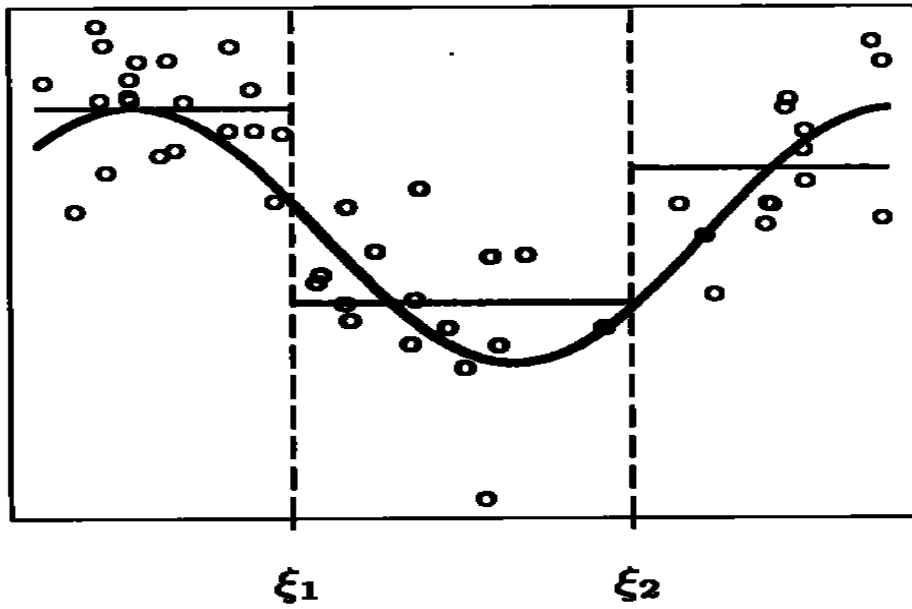
Solution: Transform $\mathbf{x} \rightsquigarrow \phi(\mathbf{x})$ with $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^p$.

Assume/hope response $y \in \mathbb{R}$ is now linear in ϕ .

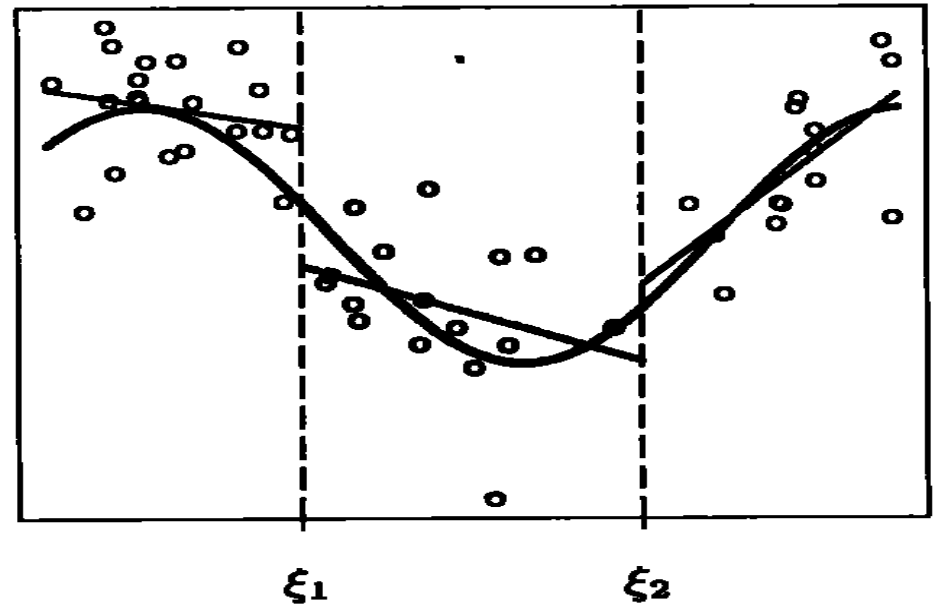
Examples:

- Linear regression: $p = d$ and $\phi_i(\mathbf{x}) = x_i$.
- Polynomial regression: $d = 1$ and $\phi_i(x) = x^i$.
- Piecewise constant regression:
E.g. $d = 1$ with $\phi_i(x) = 1$ for $i \leq x < i + 1$ and 0 else.
- Piecewise polynomials ...
- Splines ...

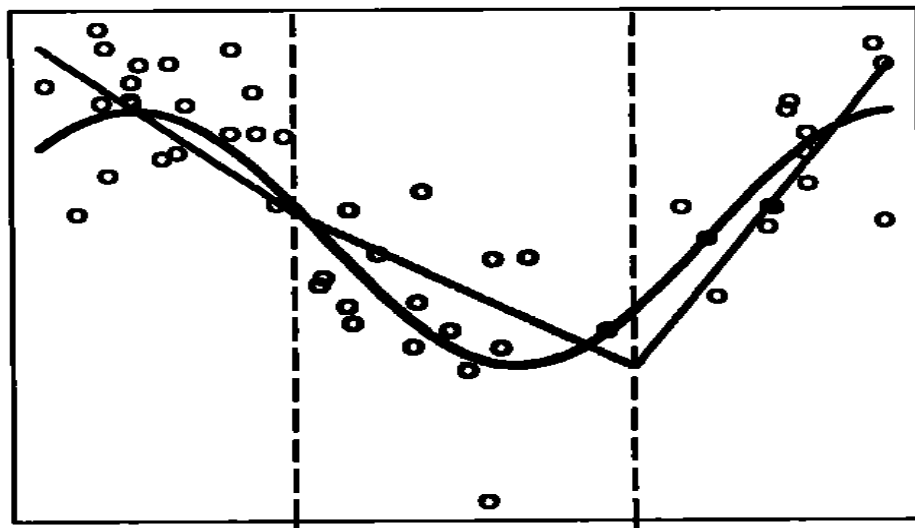
Piecewise Constant



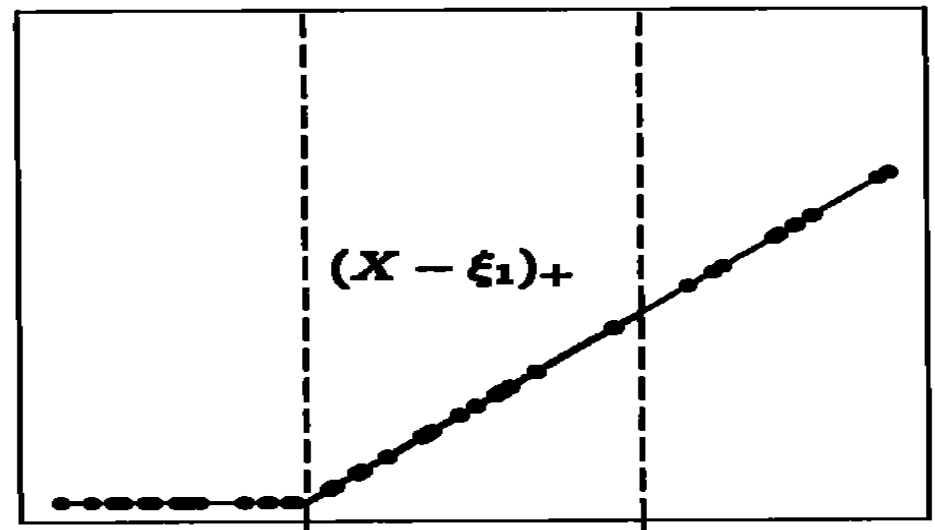
Piecewise Linear



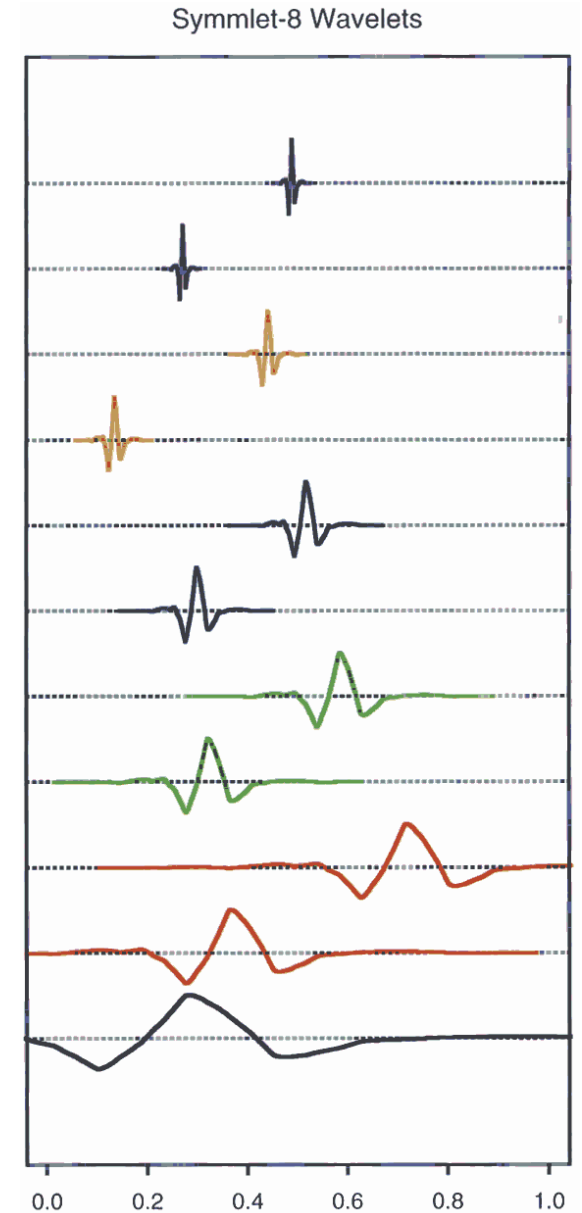
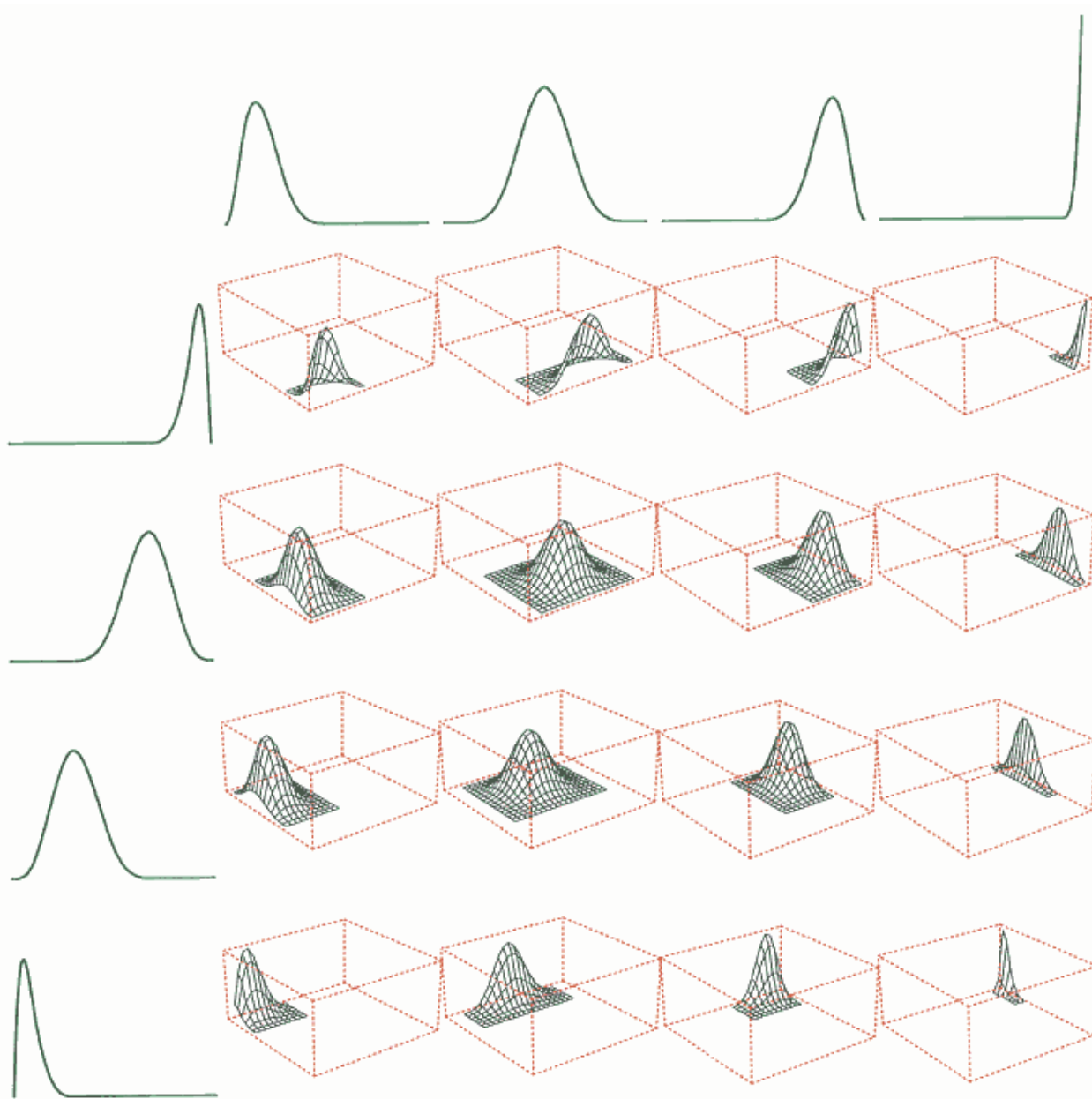
Continuous Piecewise Linear



Piecewise-linear Basis Function



2D Spline LBFR and 1D Symmlet-8 Wavelets



Local Smoothing & Kernel Regression

Estimate $f(x)$ by averaging the y_i for all x_i a -close to x :

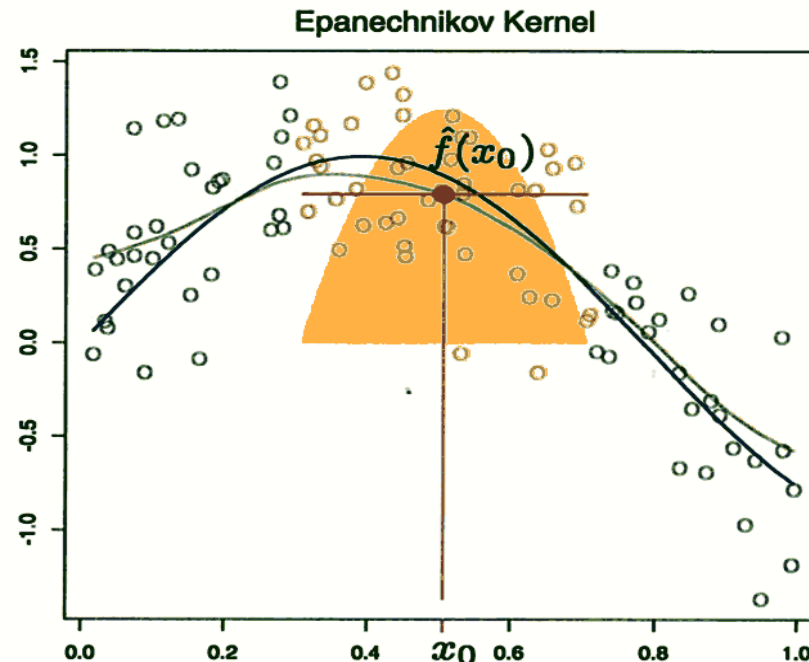
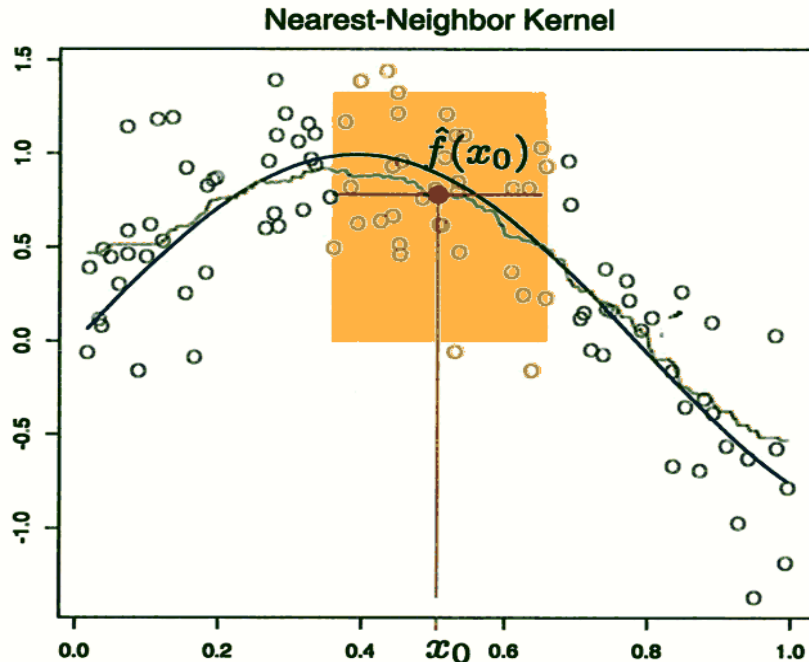
$$\hat{f}(x) = \frac{\sum_{i=1}^n K(x, x_i) y_i}{\sum_{i=1}^n K(x, x_i)}$$

Generalization to other K ,
e.g. quadratic (Epanechnikov) Kernel:

Nearest-Neighbor Kernel:

$$K(x, x_i) = 1 \text{ if } \begin{cases} |x - x_i| < a \\ \text{and } 0 \text{ else} \end{cases}$$

$$K(x, x_i) = \max\{0, a^2 - (x - x_i)^2\}$$

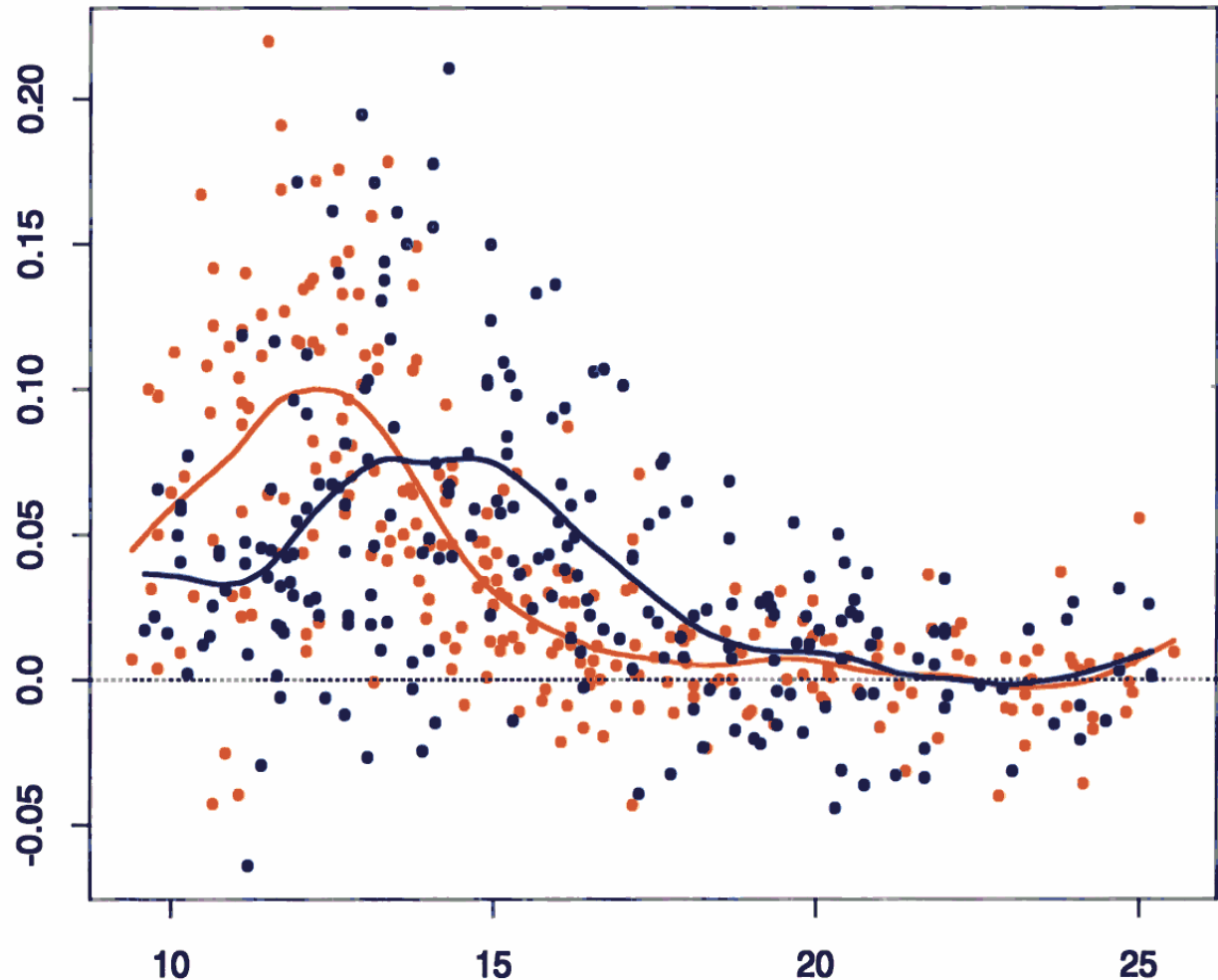


Regularization & 1D Smoothing Splines

to avoid overfitting if function class is large

$$\hat{f} = \arg \min_f \left\{ \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx \right\}$$

- $\lambda = 0$
 $\Rightarrow \hat{f} =$ any function through data
- $\lambda = \infty$
 $\Rightarrow \hat{f} =$ least squares line fit
- $0 < \lambda < \infty$
 $\Rightarrow \hat{f} =$ piecewise cubic with continuous derivative



3 NONLINEAR REGRESSION

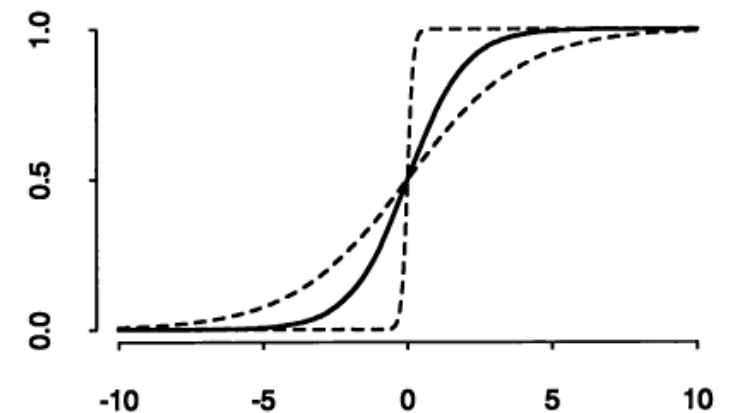
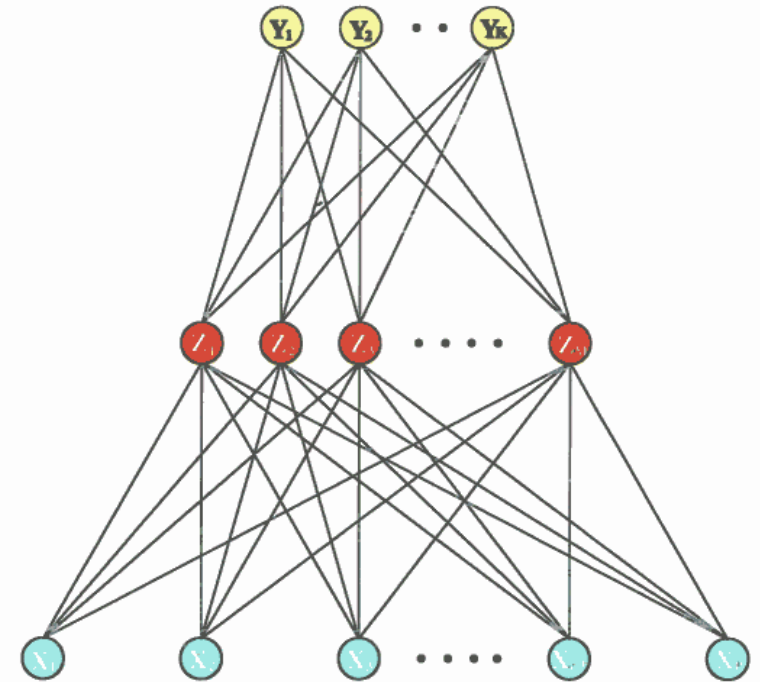
- Artificial Neural Networks
- Kernel Trick
- Maximum Margin Classifier
- Sparse Kernel Methods / SVMs

Artificial Neural Networks 1

as non-linear function approximator

Single hidden layer feed-forward neural network

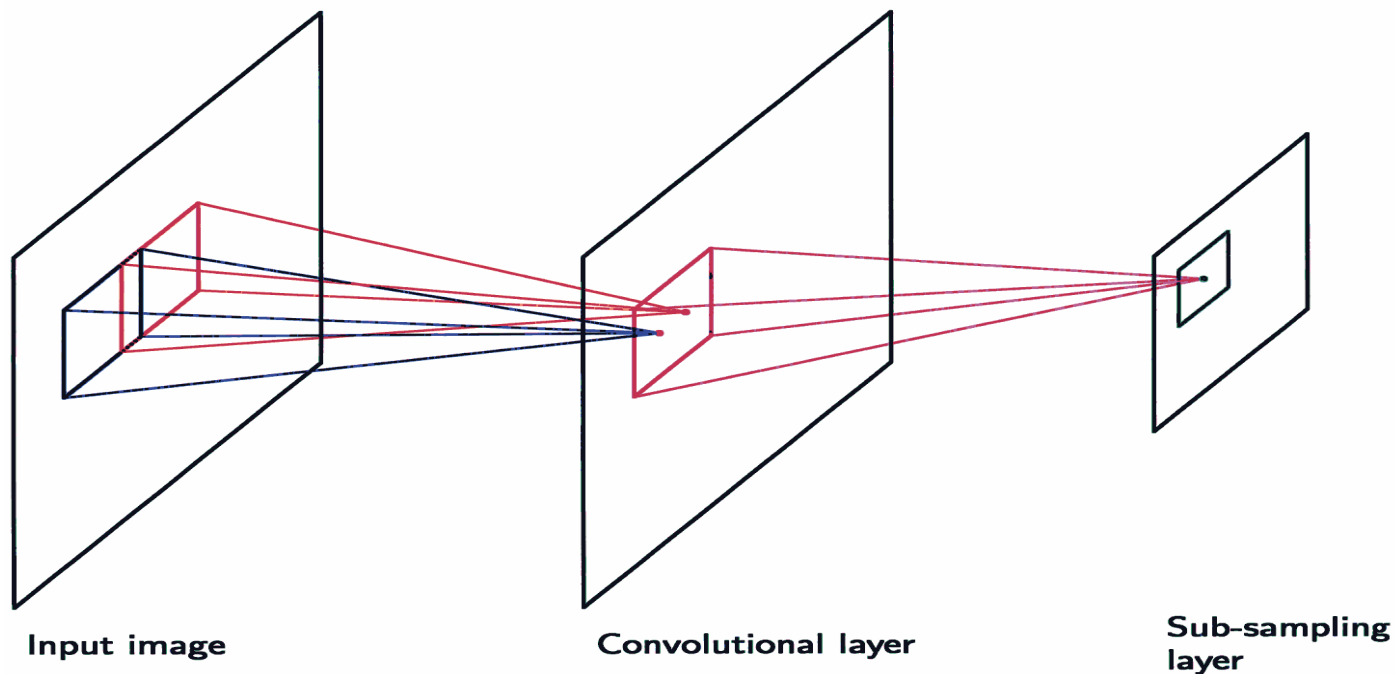
- **Hidden layer:** $z_j = h(\sum_{i=0}^d w_{ji}^{(1)} x_i)$
- **Output:** $f_{\mathbf{w},k}(\mathbf{x}) = \sigma(\sum_{j=0}^M w_{kj}^{(2)} z_j)$
- **Sigmoidal activation functions:**
 $h()$ and $\sigma()$ \Rightarrow f non-linear
- **Goal:** Find network weights best modeling the data:
- **Back-propagation algorithm:**
 Minimize $\sum_{i=1}^n \|\mathbf{y}_i - \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i)\|_2^2$
 w.r.t. \mathbf{w} by gradient decent.



Artificial Neural Networks 2

- Avoid overfitting by early stopping or small M .
- Avoid local minima by random initial weights or stochastic gradient descent.

Example: Image Processing



Kernel Trick

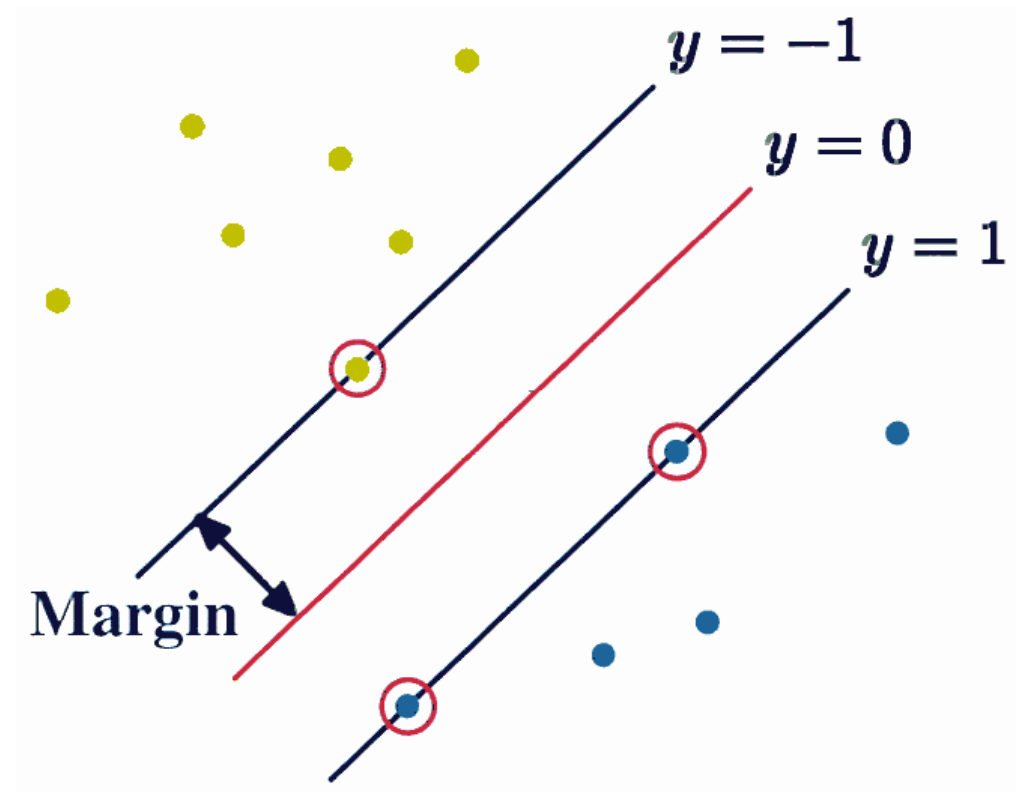
The Kernel-Trick allows to reduce the functional minimization to finite-dimensional optimization.

- Let $L(y, f(x))$ be *any* loss function
- and $J(f)$ be a penalty quadratic in f .
- then minimum of penalized loss $\sum_{i=1}^n L(y_i, f(x_i)) + \lambda J(f)$
- has form $f(x) = \sum_{i=1}^n \alpha_i K(x, x_i)$
- with α minimizing $\sum_{i=1}^n L(y_i, (\mathbf{K}\alpha)_i) + \lambda \alpha^\top \mathbf{K}\alpha$.
- and Kernel $\mathbf{K}_{ij} = K(x_i, x_j)$ following from J .

Maximum Margin Classifier

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}: \|\mathbf{w}\|=1} \min_i \{y_i(\mathbf{w}^\top \phi(x_i))\} \quad \text{with } y \in \{-1, 1\}$$

- Linear boundary for $\phi_b(\mathbf{x}) = x^{(b)}$.
- Boundary is determined by **Support Vectors** (circled data points)
- Margin negative if classes not separable.



Sparse Kernel Methods / SVMs

Non-linear boundary for general $\phi_b(x)$

$$\hat{\mathbf{w}} = \sum_{i=1}^n a_i \phi(x_i) \text{ for some } \mathbf{a}.$$

$$\Rightarrow \hat{f}(x) = \hat{\mathbf{w}}^\top \phi(x) = \sum_{i=1}^n a_i K(x_i, x)$$

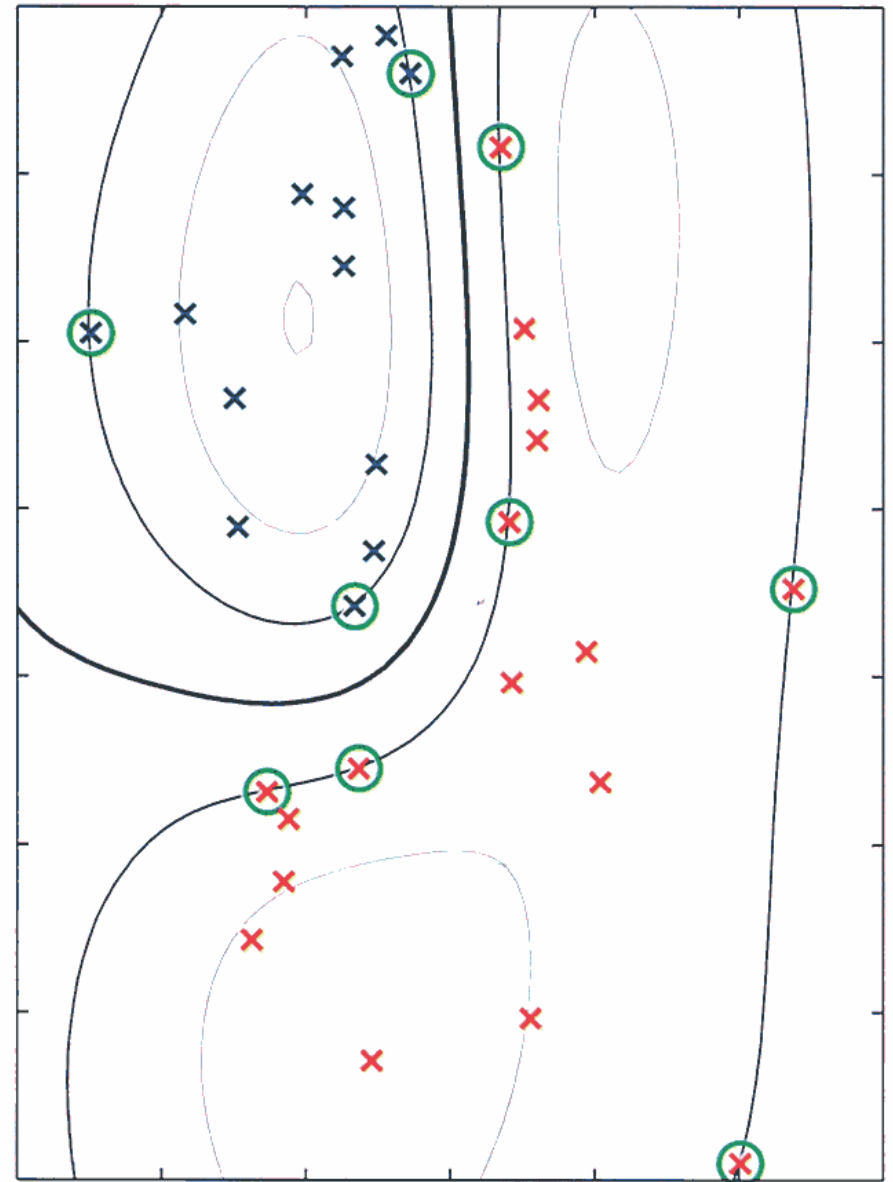
depends only on ϕ via Kernel

$$K(x_i, x) = \sum_{b=1}^d \phi_b(x_i) \phi_b(x).$$

\Rightarrow Huge time savings if $d \gg n$

Example $K(x, x')$:

- polynomial $(1 + \langle x, x' \rangle)^d$,
- Gaussian $\exp(-\|x - x'\|_2^2)$,
- neural network $\tanh(\langle x, x' \rangle)$.

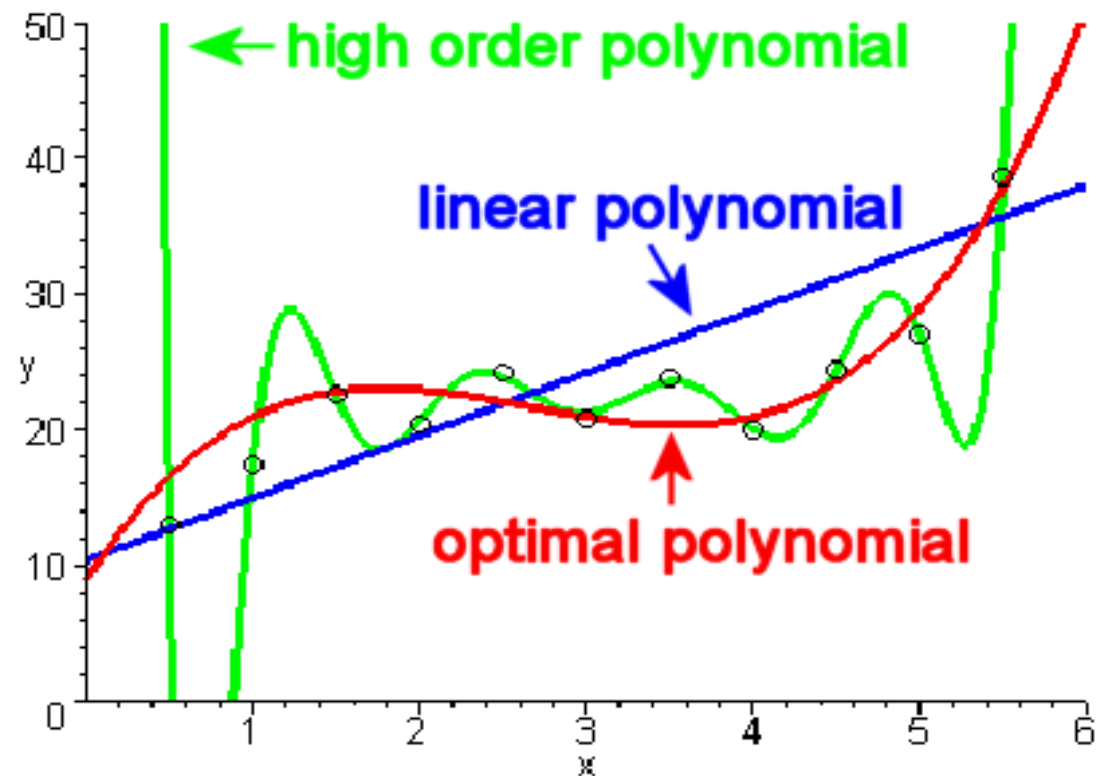


4 MODEL ASSESSMENT & SELECTION

- Example: Polynomial Regression
- Training=Empirical versus Test=True Error
- Empirical Model Selection
- Theoretical Model Selection
- The Loss Rank Principle for Model Selection

Example: Polynomial Regression

- Straight line does not fit data well (large training error)
high bias \Rightarrow poor predictive performance
- High order polynomial fits data perfectly (zero training error)
high variance (overfitting)
 \Rightarrow poor prediction too!
- Reasonable polynomial degree d performs well.
How to select d ?
minimizing training error obviously does not work.



Training=Empirical versus Test=True Error

- Learn functional relation f for data $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$.
- We can compute the **empirical error** on past data:
$$\text{Err}_D(f) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2.$$
- Assume data D is sample from some distribution P .
- We want to know the expected **true error** on future examples:
$$\text{Err}_P(f) = \mathbf{E}_P[(y - f(x))].$$
- How good an estimate of $\text{Err}_P(f)$ is $\text{Err}_D(f)$?
- **Problem:** $\text{Err}_D(f)$ decreases with increasing model complexity, but not $\text{Err}_P(f)$.

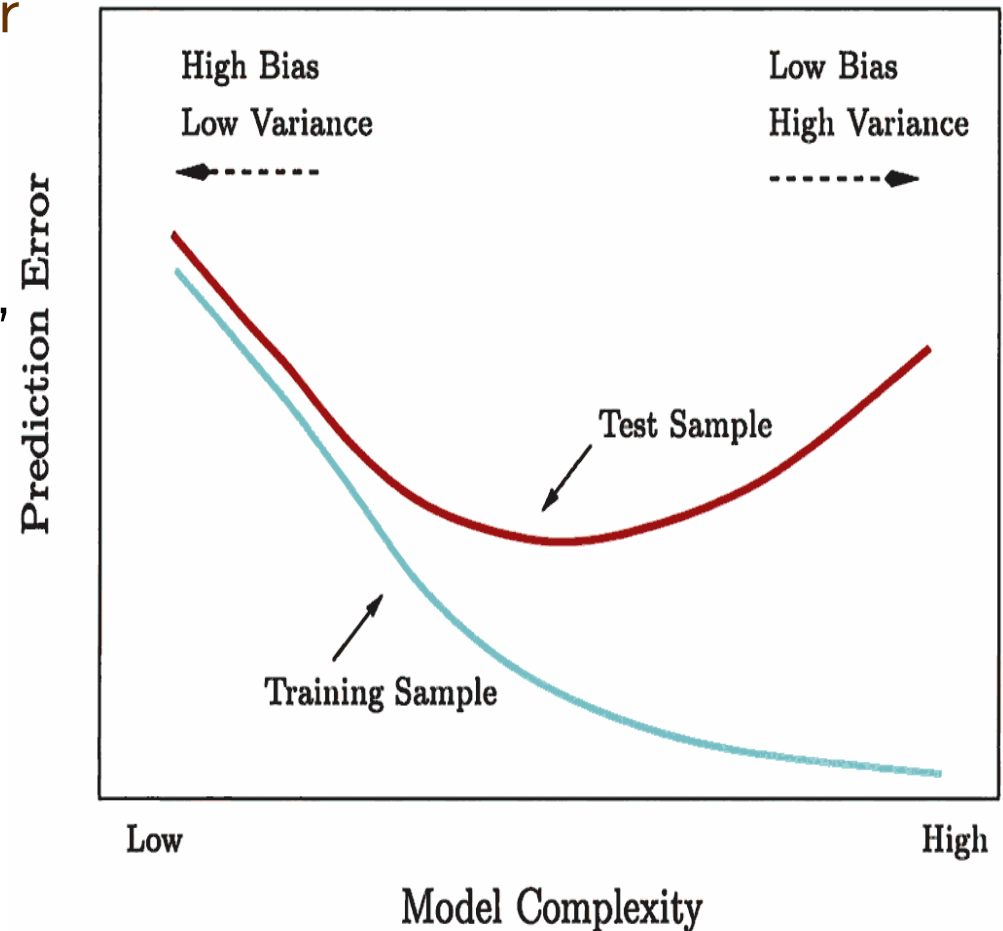
Empirical Model Selection

How to select complexity parameter

- Kernel width a ,
- penalization constant λ ,
- number k of nearest neighbors,
- the polynomial degree d ?

Empirical test-set-based methods:

Regress on training set and minimize empirical error w.r.t. “complexity” parameter (a, λ, k, d) on a separate test-set.



Sophistication: cross-validation, bootstrap, ...

Theoretical Model Selection

How to select complexity or flexibility or smoothness parameter:

Kernel width a , penalization constant λ , number k of nearest neighbors, the polynomial degree d ?

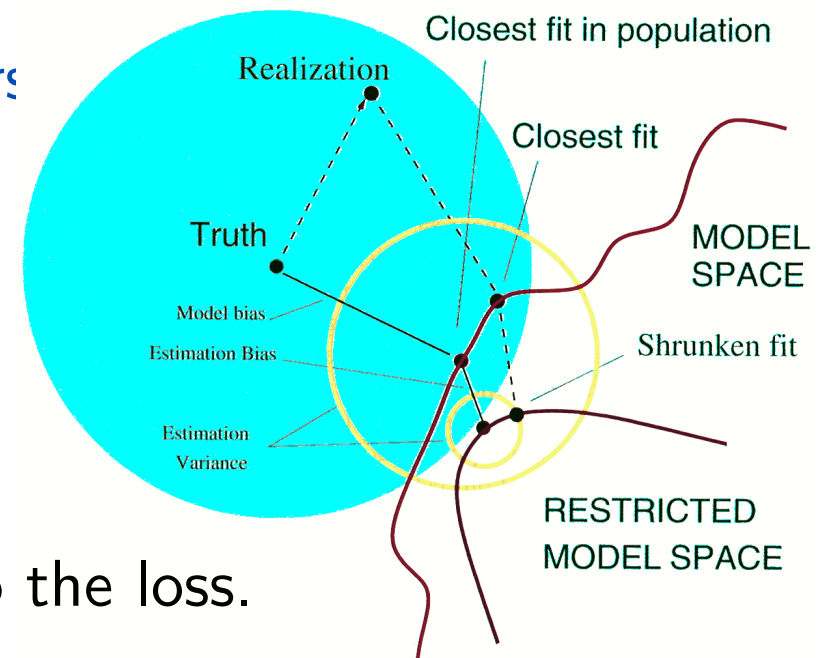
For parametric regression with d parameters:

- Bayesian model selection,
- Akaike Information Criterion (AIC),
- Bayesian Information Criterion (BIC),
- Minimum Description Length (MDL),

They all add a penalty proportional to d to the loss.

For non-parametric linear regression:

- Add trace of on-data regressor = effective $\#$ of parameters to loss.
- Loss Rank Principle (LoRP).



The Loss Rank Principle for Model Selection

Let $\hat{f}_D^c : \mathcal{X} \rightarrow \mathcal{Y}$ be the (best) regressor of complexity c on data D .

The loss *Rank* of \hat{f}_D^c is defined as the number of other (fictitious) data D' that are fitted better by \hat{f}_D^c , than D is fitted by \hat{f}_D^c .

- c is small $\Rightarrow \hat{f}_D^c$ fits D badly \Rightarrow many other D' can be fitted better \Rightarrow *Rank* is large.
- c is large \Rightarrow many D' can be fitted well \Rightarrow *Rank* is large.
- c is appropriate $\Rightarrow \hat{f}_D^c$ fits D well *and* not too many other D' can be fitted well \Rightarrow *Rank* is small.

LoRP: Select model complexity c that has minimal loss *Rank*

Unlike most penalized maximum likelihood variants (AIC, BIC, MDL),

- LoRP only depends on the regression and the loss function.
- It works without a stochastic noise model, and
- is directly applicable to any non-parametric regressor, like kNN.

5 HOW TO ATTACK LARGE PROBLEMS

- Probabilistic Graphical Models (PGM)
- Trees Models
- Non-Parametric Learning
- Approximate (Variational) Inference
- Sampling Methods
- Combining Models
- Boosting

Probabilistic Graphical Models (PGM)

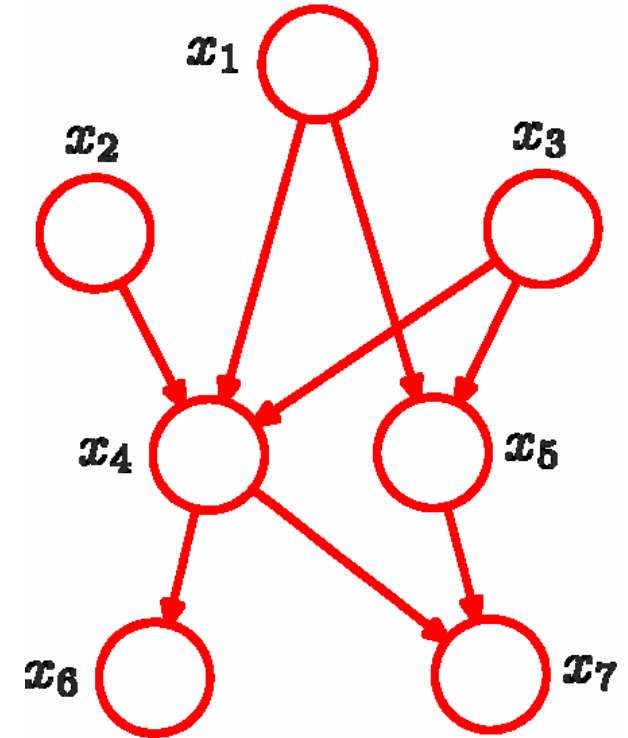
Visualize structure of model and (in)dependence

⇒ faster and more comprehensible algorithms

- **Nodes** = random variables.
- **Edges** = stochastic dependencies.
- **Bayesian network** = directed PGM
- **Markov random field** = undirected PGM

Example:

$$P(x_1)P(x_2)P(x_3)P(x_4|x_1x_2x_3)P(x_5|x_1x_3)P(x_6|x_4)P(x_7|x_4x_5)$$

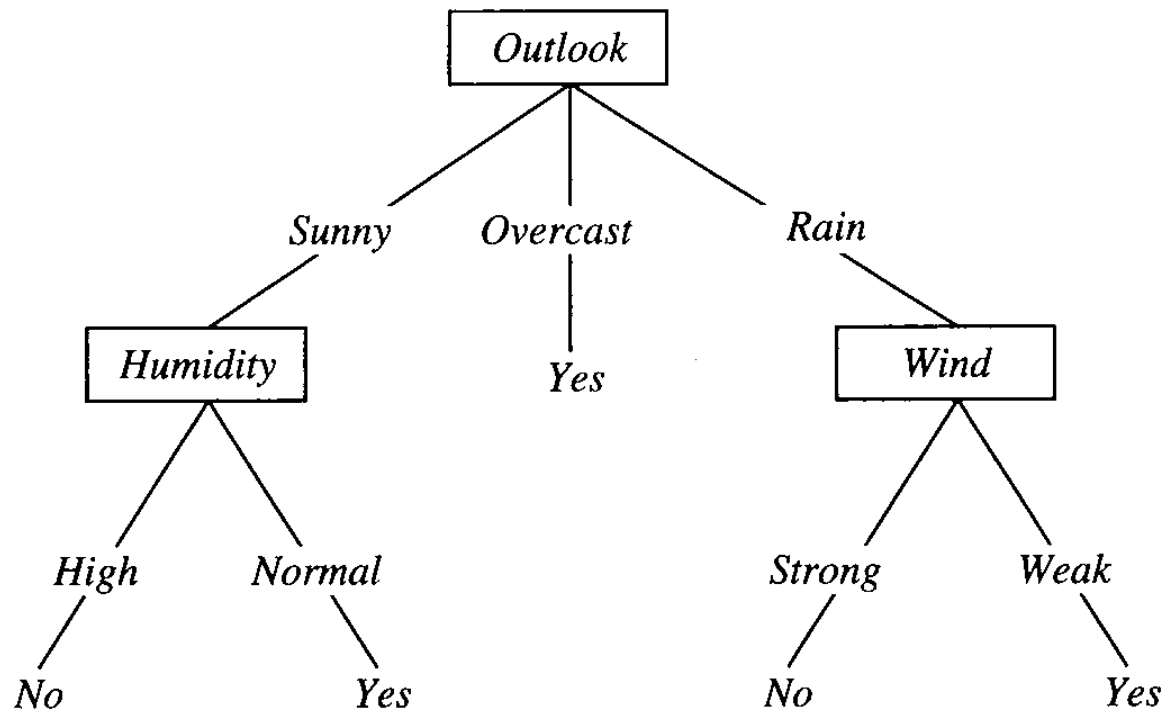


Additive Models & Trees & Related Methods

Generalized additive model: $f(\mathbf{x}) = \alpha + f_1(x_1) + \dots + f_d(x_d)$

Reduces determining $f : \mathbb{R}^d \rightarrow \mathbb{R}$ to d 1d functions $f_b : \mathbb{R} \rightarrow \mathbb{R}$

Classification/decision trees:

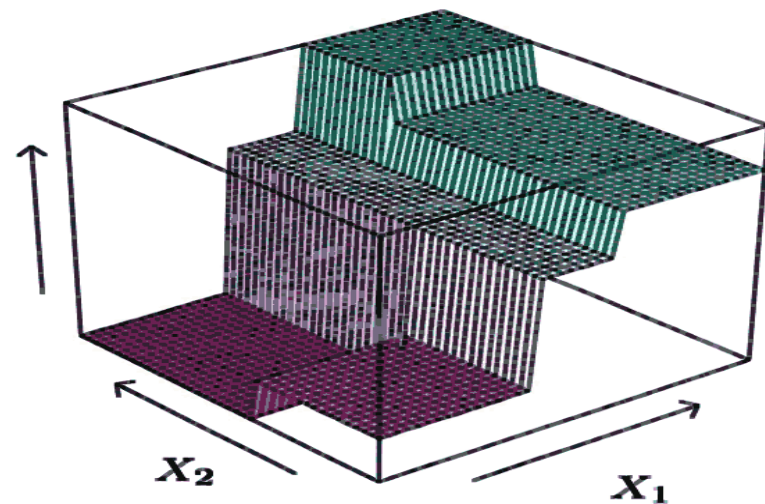
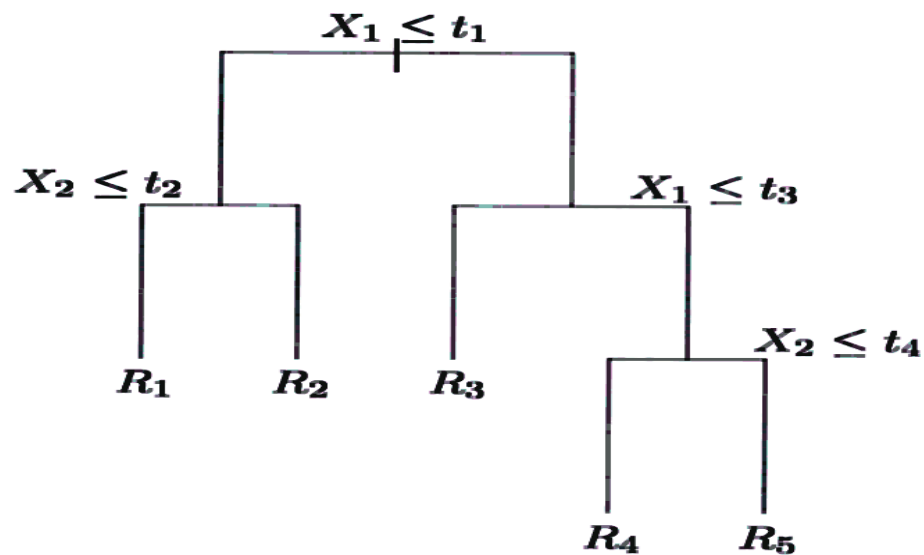
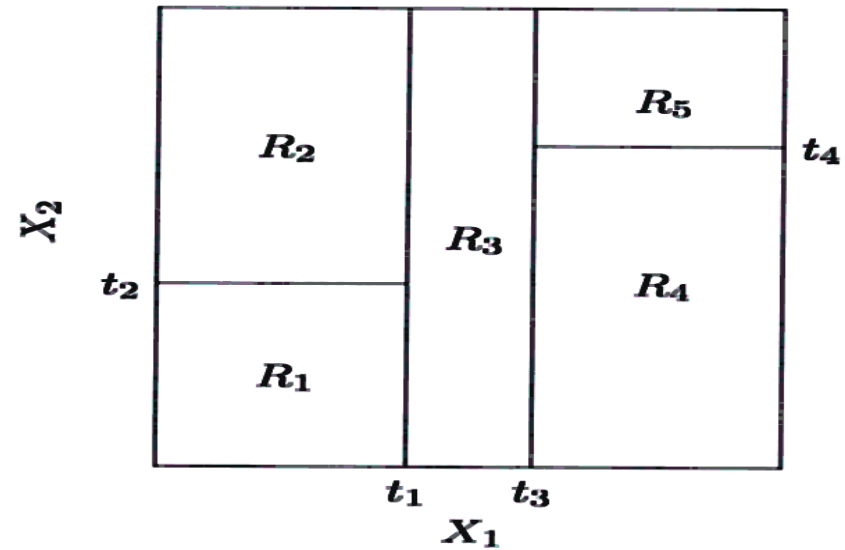
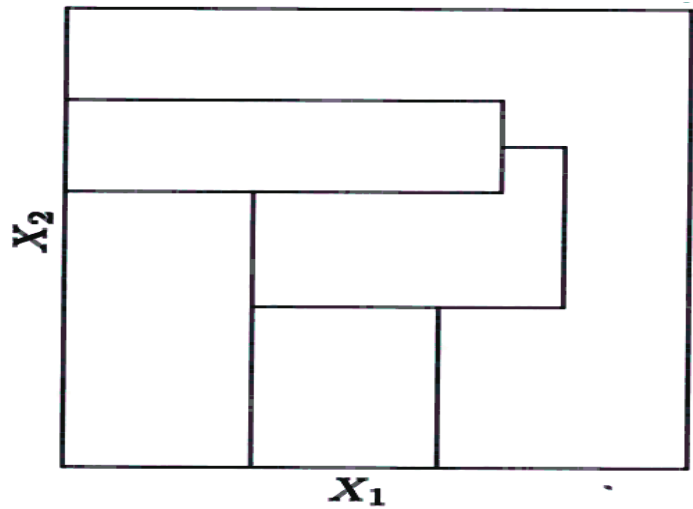


Outlook:

- PRIM,
- bump hunting,
- How to **learn** tree structures.

Regression Trees

$$f(x) = c_b \text{ for } x \in R_b, \text{ and } c_b = \text{Average}[y_i | x_i \in R_b]$$



Non-Parametric Learning

= prototype methods = instance-based learning = model-free

Examples:

- **K-means:**

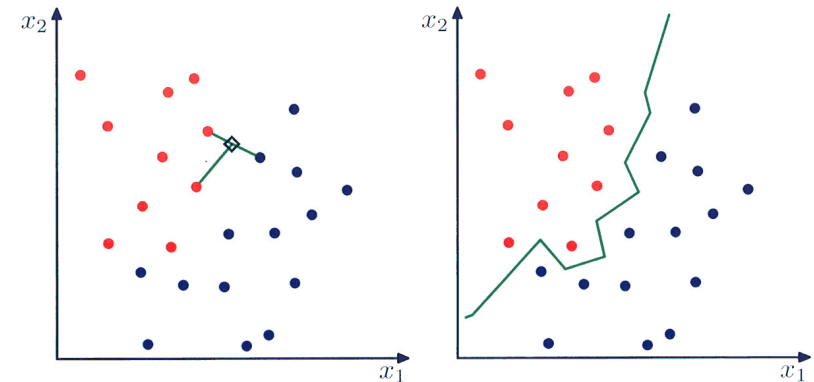
Data clusters around K centers with cluster means μ_1, \dots, μ_K .

Assign x_i to closest cluster center.

- **K Nearest neighbors regression (kNN):**

Estimate $f(x)$ by averaging the y_i

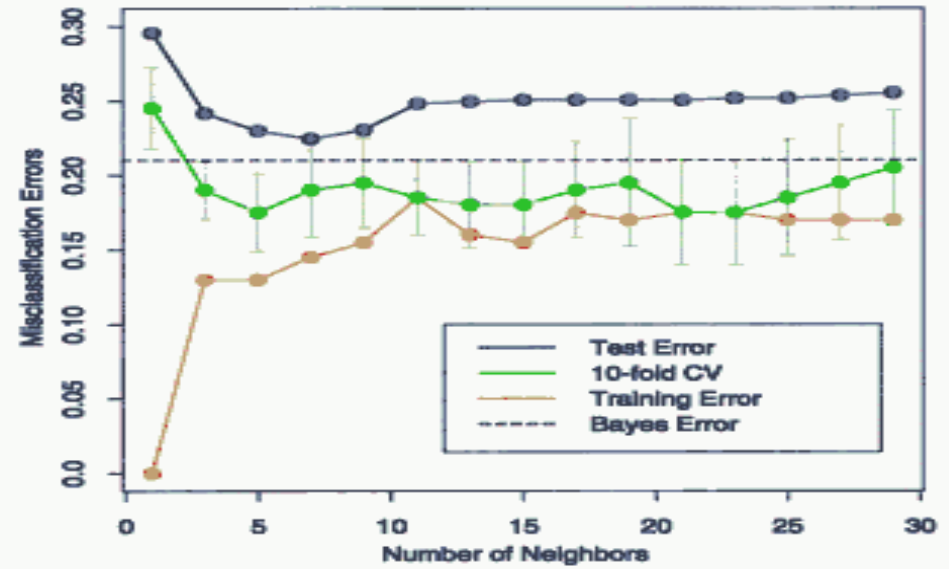
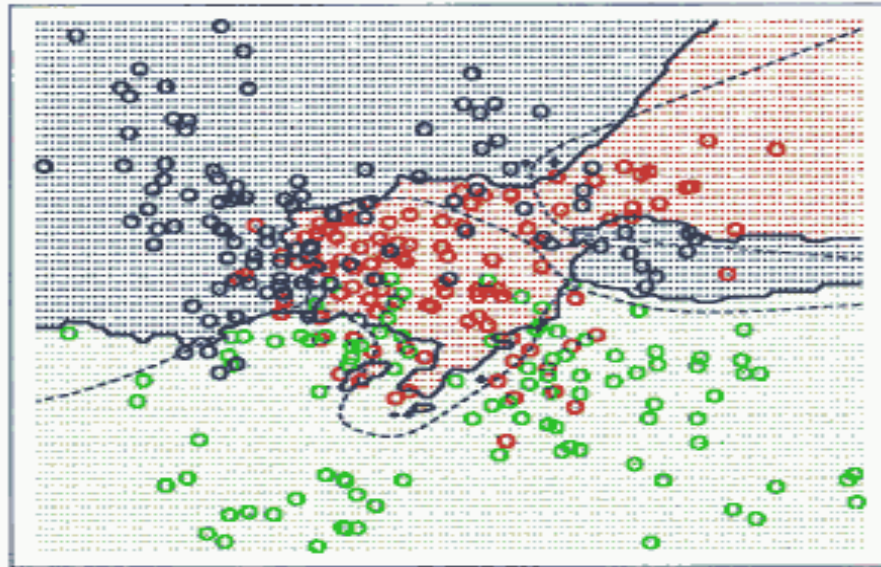
for the k x_i closest to x :



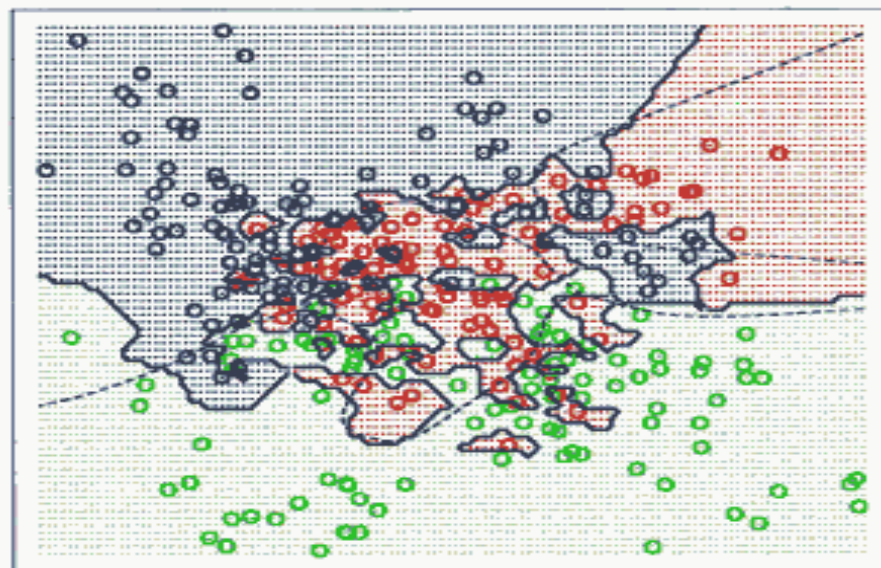
- **Kernel regression:**

Take a weighted average $\hat{f}(x) = \frac{\sum_{i=1}^n K(x, x_i) y_i}{\sum_{i=1}^n K(x, x_i)}$.

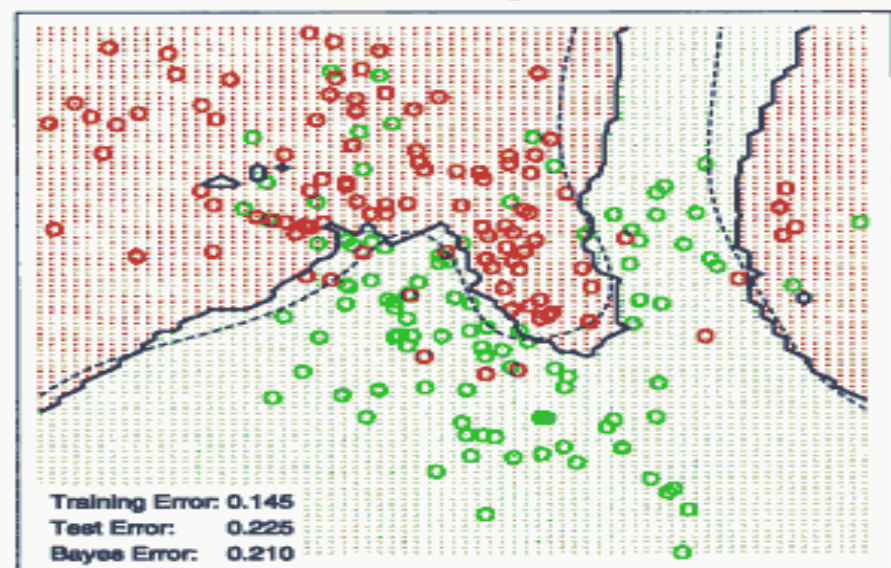
15-Nearest Neighbors



1-Nearest Neighbor



7-Nearest Neighbors



Approximate (Variational) Inference

Approximate full distribution $P(\mathbf{z})$ by $q(\mathbf{z})$

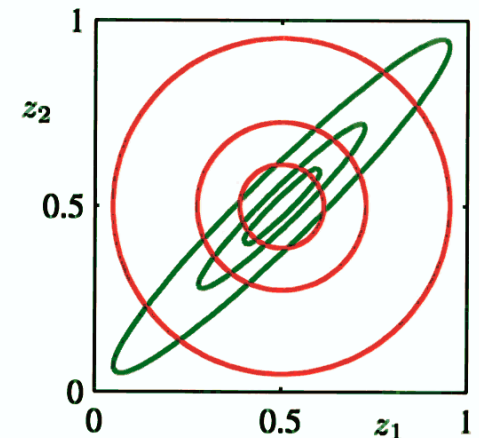
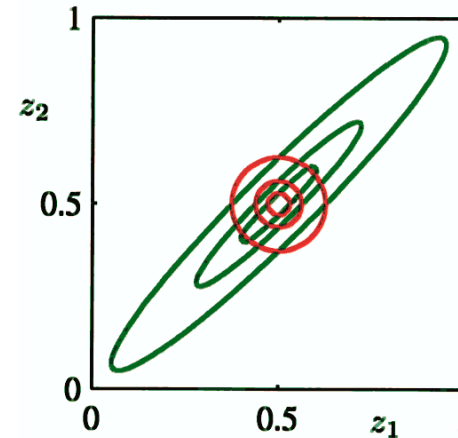
Examples: Gaussians

Popular: Factorized distribution:

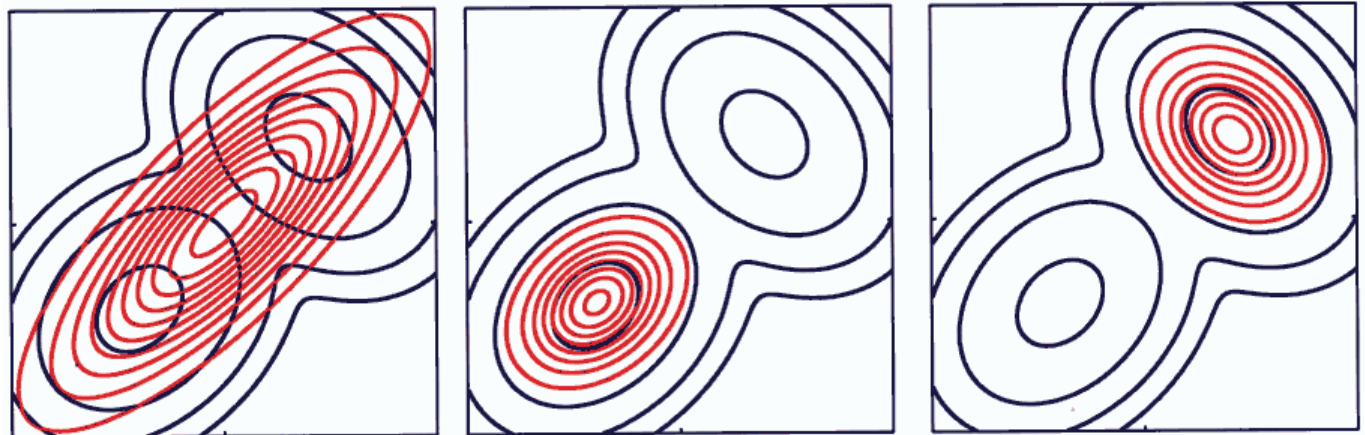
$$q(\mathbf{z}) = q_1(z_1) \times \dots \times q_M(z_M).$$

Measure of fit: Relative entropy

$$\text{KL}(p||q) = \int p(\mathbf{z}) \log \frac{p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z}$$



Red curves: Left minimizes $\text{KL}(P||q)$, Middle and Right are the two local minima of $\text{KL}(q||P)$.



Elementary Sampling Methods

How to sample from $P : \mathcal{Z} \rightarrow [0, 1]$?

- **Special sampling algorithms** for standard distributions P .
- **Rejection sampling**: Sample z uniformly from domain \mathcal{Z} , but accept sample only with probability $\propto P(z)$.

- **Importance sampling**:

$$\mathbf{E}[f] = \int f(z)p(z)dz \simeq \frac{1}{L} \sum_{l=1}^L f(z_l)p(z_l)/q(z_l),$$

where z_l are sampled from q .

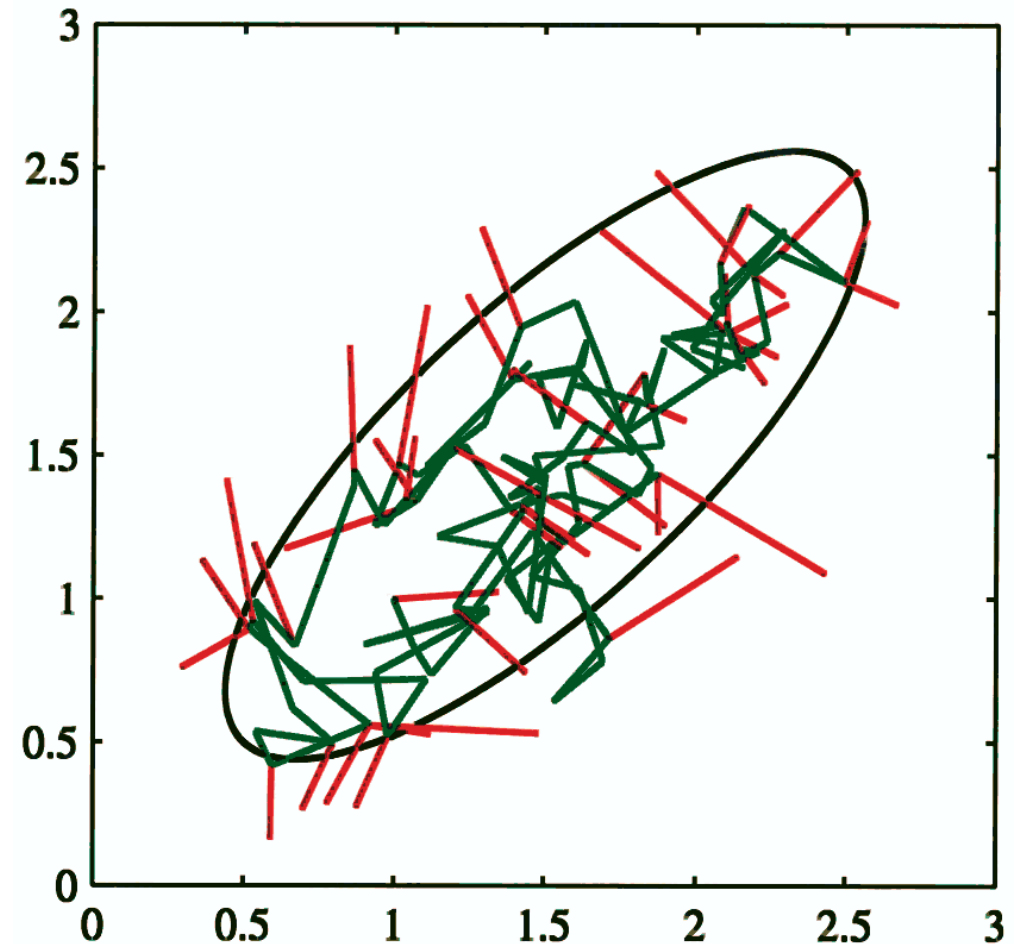
Choose q easy to sample and large where $f(z_l)p(z_l)$ is large.

- **Others**: **Slice sampling**

Markov Chain Monte Carlo (MCMC) Sampling

Metropolis: Choose some convenient q with $q(\mathbf{z}|\mathbf{z}') = q(\mathbf{z}'|\mathbf{z})$. Sample \mathbf{z}_{l+1} from $q(\cdot|\mathbf{z}_l)$ but accept only with probability $\min\{1, p(\mathbf{z}_{l+1})/p(\mathbf{z}_l)\}$.

Gibbs Sampling: Metropolis with q leaving \mathbf{z} unchanged from $l \rightsquigarrow l+1$, except resample coordinate i from $P(z_i|\mathbf{z}_{\setminus i})$.



Green lines are accepted and red lines are rejected Metropolis steps.

Combining Models

**Performance can often be improved
by combining multiple models in some way,
instead of just using a single model in isolation**

- **Committees:** Average the predictions of a set of individual models.
- **Bayesian model averaging:** $P(x) = \sum_{\text{Models}} P(x|\text{Model})P(\text{Model})$
- **Mixture models:** $P(x|\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_k \pi_k P_k(x|\boldsymbol{\theta}_k)$
- **Decision tree:** Each model is responsible for a different part of the input space.
- **Boosting:** Train many weak classifiers in sequence and combine output to produce a powerful committee.

Boosting

Idea: Train many weak classifiers G_m in sequence and combine output to produce a powerful committee G .

AdaBoost.M1: [Freund & Schapire (1997) received famous Gödel-Prize]

Initialize observation weights w_i uniformly.

For $m = 1$ to M do:

(a) G_m classifies x as $G_m(x) \in \{-1, 1\}$.

Train G_m weighing data i with w_i .

(b) Give G_m high/low weight α_i if it performed well/bad.

(c) Increase attention=weight w_i for obs. x_i misclassified by G_m .

Output weighted majority vote: $G(x) = \text{sign}(\sum_{m=1}^M \alpha_m G_m(x))$

6 UNSUPERVISED LEARNING

- K-Means Clustering
- Mixture Models
- Expectation Maximization Algorithm

Unsupervised Learning

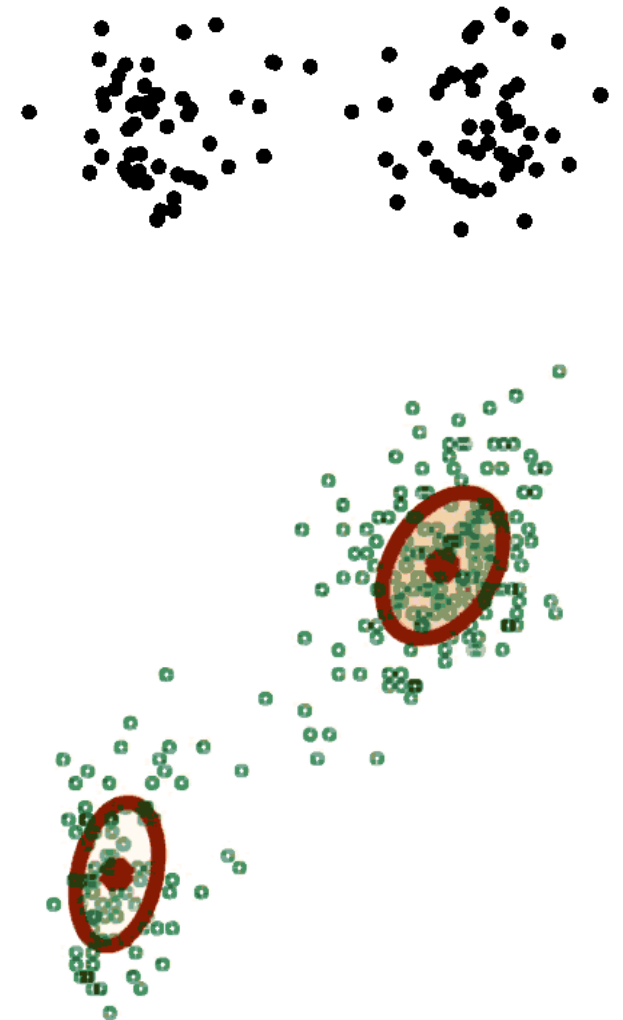
Supervised learning: Find functional relationship $f : \mathcal{X} \rightarrow \mathcal{Y}$ from I/O data $\{(x_i, y_i)\}$

Unsupervised learning:
Find pattern in data (x_1, \dots, x_n)
without an explicit teacher (no y values).

Example: Clustering e.g. by K-means

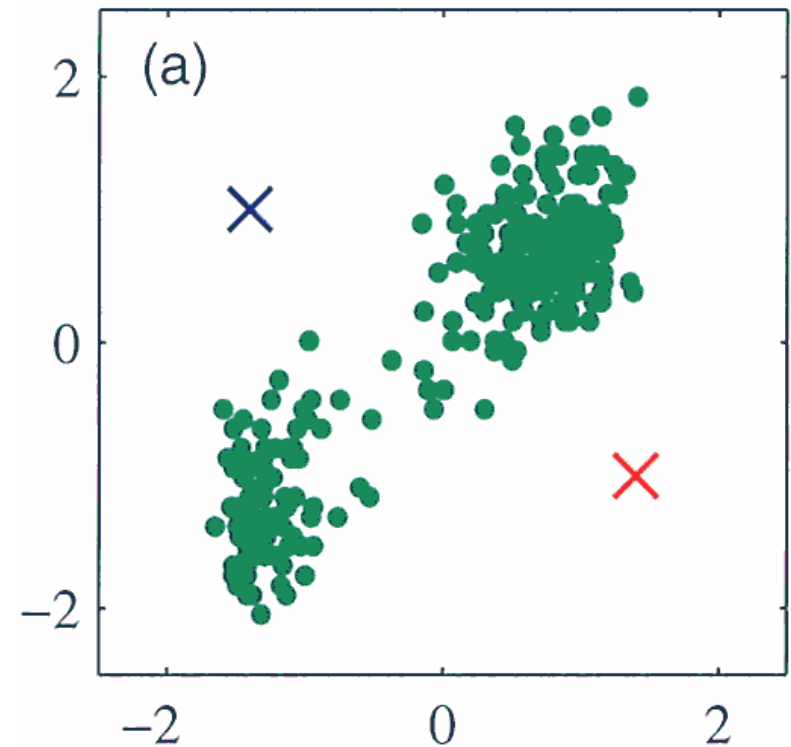
Implicit goal: Find simple explanation,
i.e. compress data (MDL, Occam's razor).

Density estimation: From which probability
distribution P are the x_i drawn?

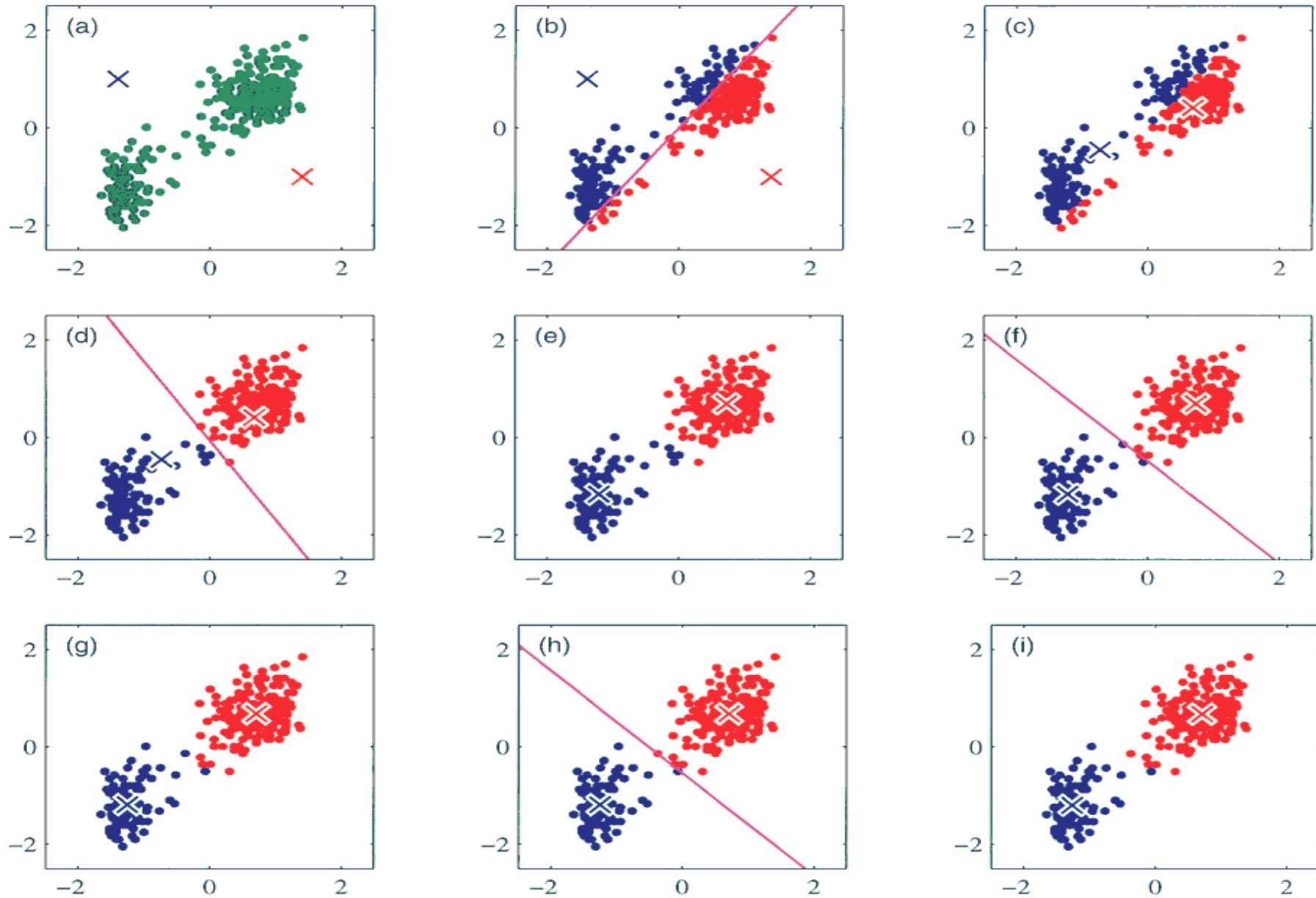


K-means Clustering and EM

- Data points seem to cluster around two centers.
- Assign each data point i to a cluster k_i .
- Let $\mu_k =$ center of cluster k .
- **Distortion measure:** Total distance² of data points from cluster centers:
$$J(\mathbf{k}, \boldsymbol{\mu}) := \sum_{i=1}^n \|x_i - \mu_{k_i}\|^2$$
- Choose centers μ_k initially at random.
- **M-step:** Minimize J w.r.t. \mathbf{k} :
Assign each point to closest center
- **E-step:** Minimize J w.r.t. $\boldsymbol{\mu}$:
Let μ_k be the mean of points belonging to cluster k
- **Iteration** of M-step and E-step converges to local minimum of J .



Iterations of K-means EM Algorithm

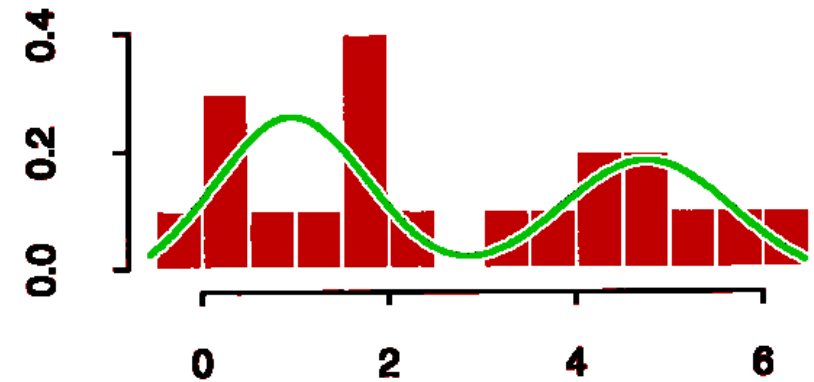


Mixture Models and EM

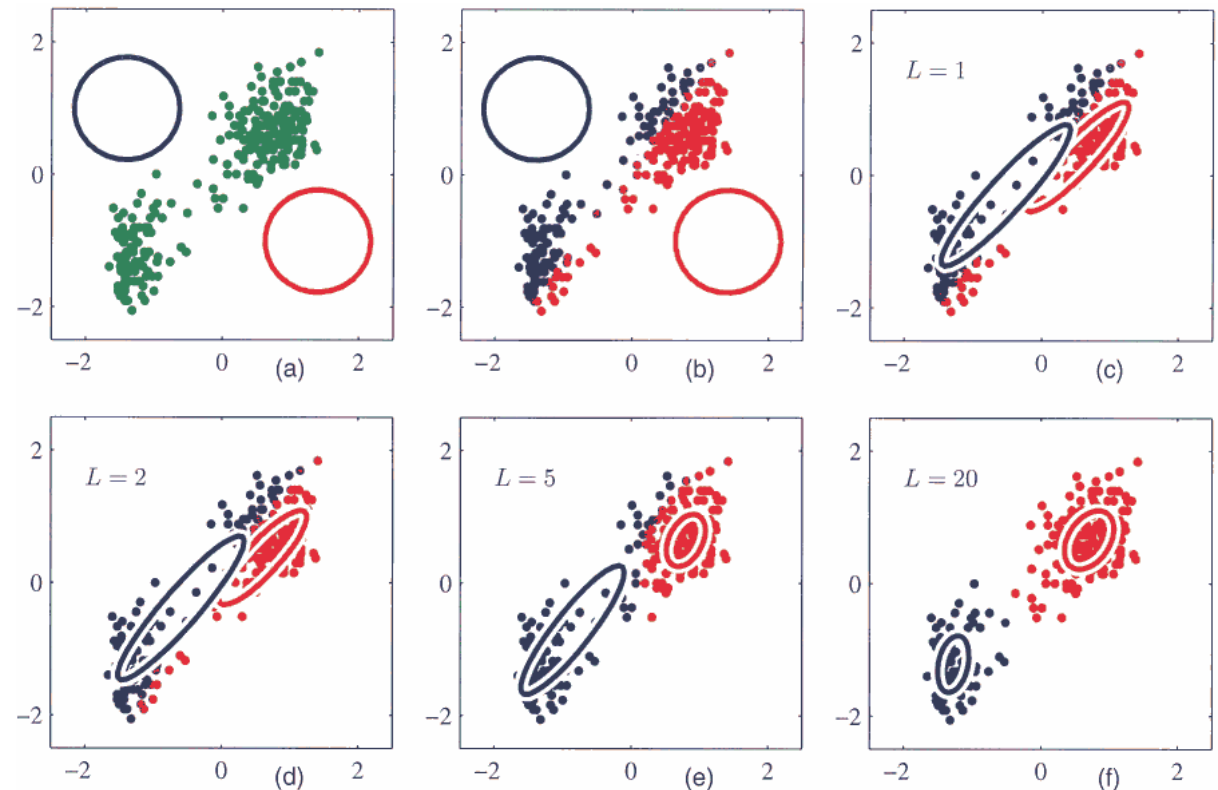
Mixture of Gaussians:

$$P(\mathbf{x}|\boldsymbol{\pi}\boldsymbol{\mu}\boldsymbol{\Sigma}) = \sum_{i=1}^K \text{Gauss}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k$$

Maximize likelihood $P(\mathbf{x}|\dots)$ w.r.t. $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.



E-Step: Compute probability γ_{ik} that data point i belongs to cluster k , based on estimates $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.



M-Step: Re-estimate $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ (take empirical mean/variance) given γ_{ik} .

7 NON-IID: SEQUENTIAL & (RE)ACTIVE SETTINGS

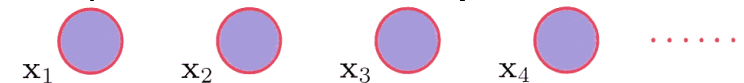
- Sequential Data
- Sequential Decision Theory
- Learning Agents
- Reinforcement Learning (RL)
- Learning in Games

Sequential Data

General stochastic time-series: $P(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | x_1 \dots x_{i-1})$

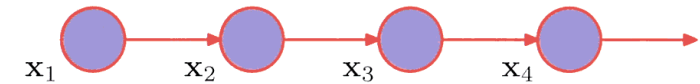
Independent identically distributed (roulette, dice, classification):

$$P(x_1 \dots x_n) = P(x_1)P(x_2) \dots P(x_n)$$



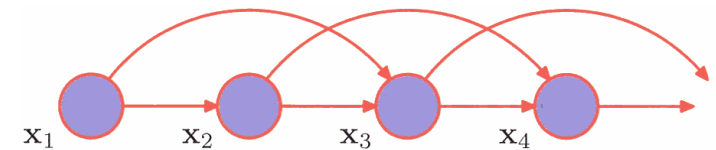
First order Markov chain (Backgammon):

$$P(x_1 \dots x_n) = P(x_1)P(x_2 | x_1)P(x_3 | x_2) \dots P(x_n | x_{n-1})$$



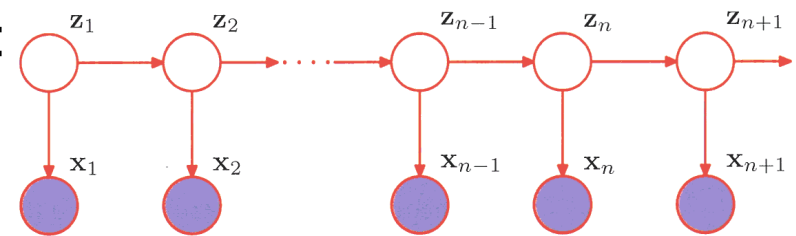
Second order Markov chain (mechanical systems):

$$P(x_1 \dots x_n) = P(x_1)P(x_2 | x_1)P(x_3 | x_1 x_2) \times \\ \dots \times P(x_n | x_{n-1} x_{n-2})$$



Hidden Markov model (speech recognition):

$$P(x_1 \dots x_n) = \int P(z_1)P(z_2 | z_1) \dots P(z_n | z_{n-1}) \\ \times \prod_{i=1}^n P(x_i | z_i) dz$$



Sequential Decision Theory

Setup:

For $t = 1, 2, 3, 4, \dots$

Given sequence x_1, x_2, \dots, x_{t-1}

(1) predict/make decision y_t ,

(2) observe x_t ,

(3) suffer loss $\text{Loss}(x_t, y_t)$,

(4) $t \rightarrow t + 1$, goto (1)

Example: Weather Forecasting

$x_t \in \mathcal{X} = \{\text{sunny, rainy}\}$

$y_t \in \mathcal{Y} = \{\text{umbrella, sunglasses}\}$

Loss	sunny	rainy
umbrella	0.1	0.3
sunglasses	0.0	1.0

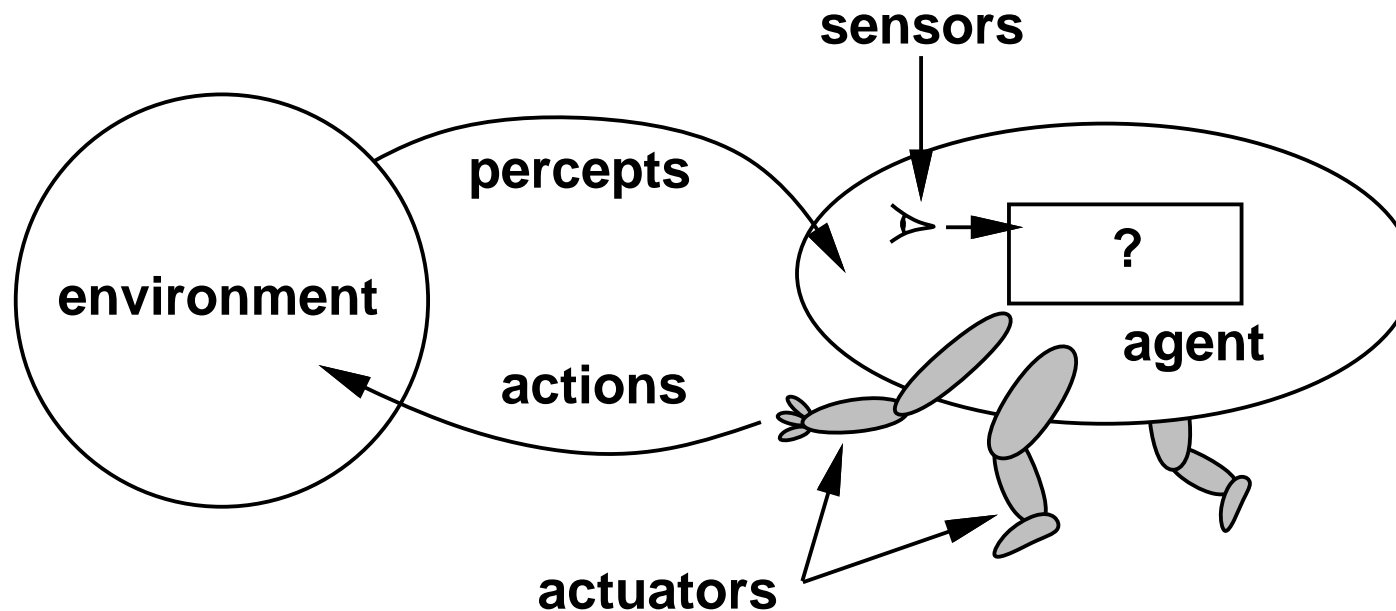
Goal: Minimize expected Loss: $\hat{y}_t = \arg \min_{y_t} \mathbf{E}[\text{Loss}(x_t, y_t) | x_1 \dots x_{t-1}]$

Greedy minimization of expected loss **is optimal** if:

Important: Decision y_t does not influence env. (future observations).

Examples: Loss = square / absolute / 0-1 error function
 \hat{y} = mean / median / mode of $P(x_t | \dots)$

Learning Agents 1



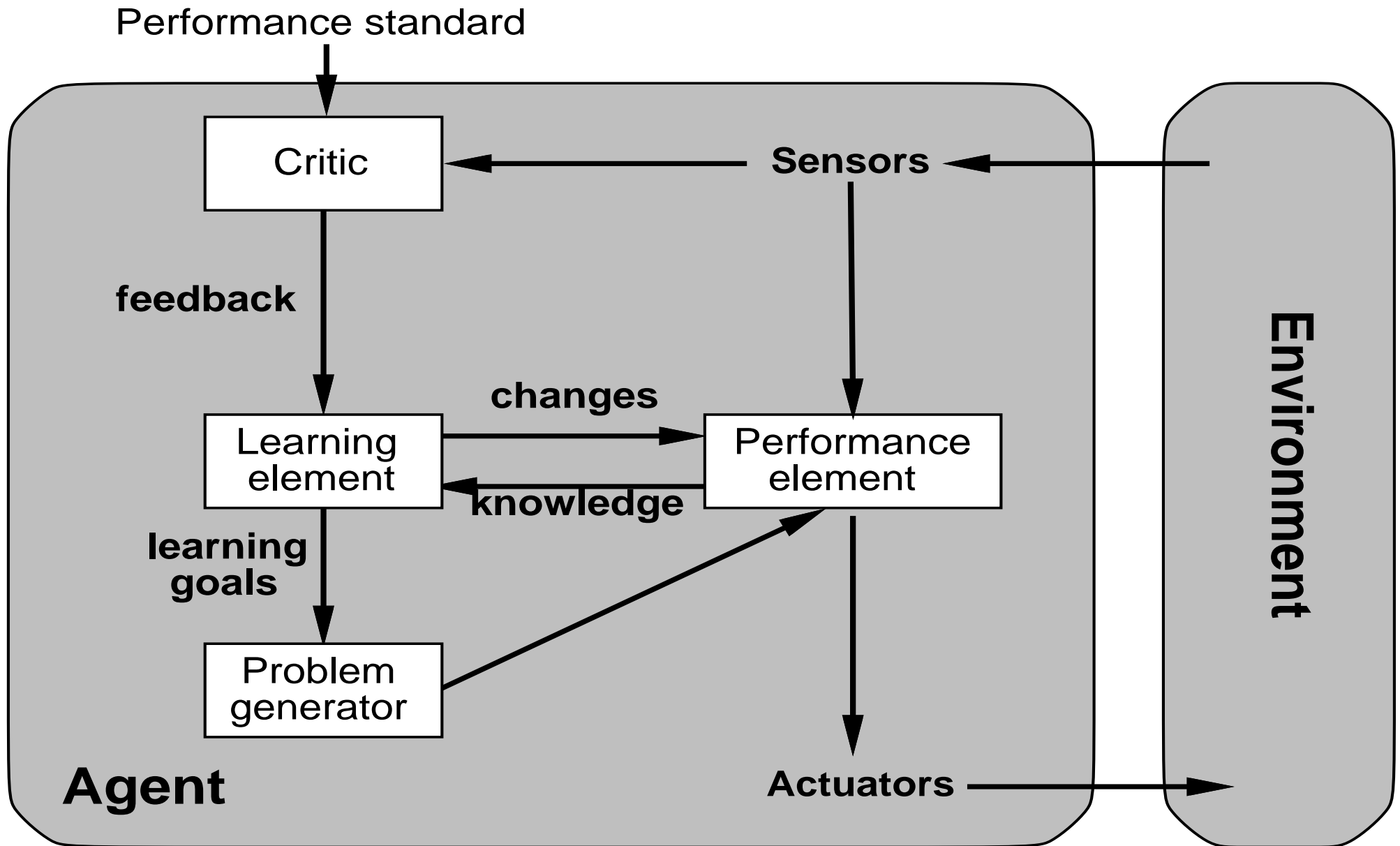
Additional complication:

Learner can influence environment,
and hence what he observes next.

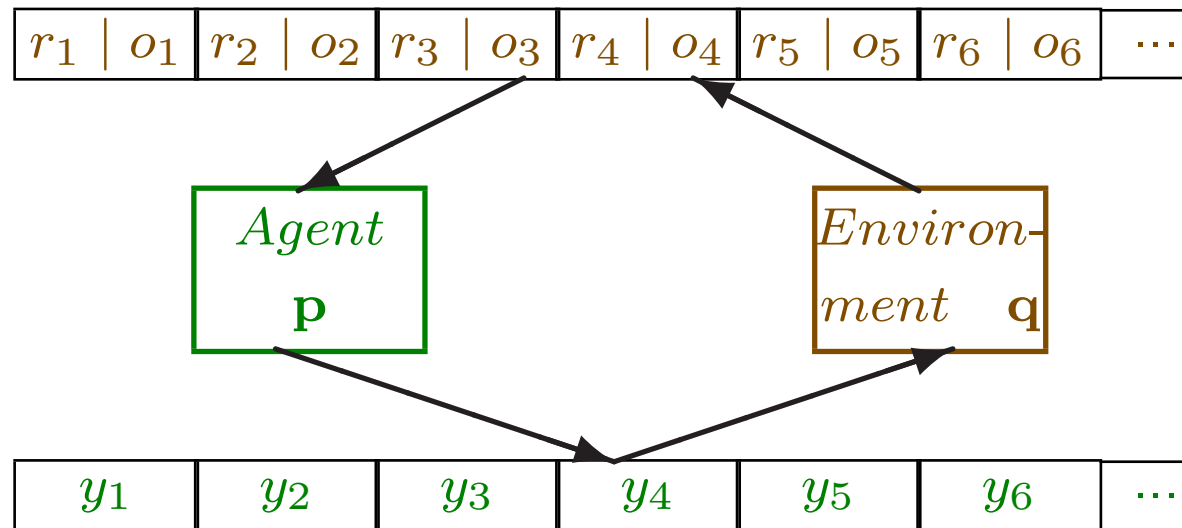
⇒ farsightedness, planning, exploration necessary.

Exploration ⇔ Exploitation problem

Learning Agents 2



Reinforcement Learning (RL)



- RL is concerned with how an agent ought to take actions in an environment so as to maximize its long-term reward.
- Find policy that maps states of the world to the actions the agent ought to take in those states.
- The environment is typically formulated as a finite-state Markov decision process (MDP).

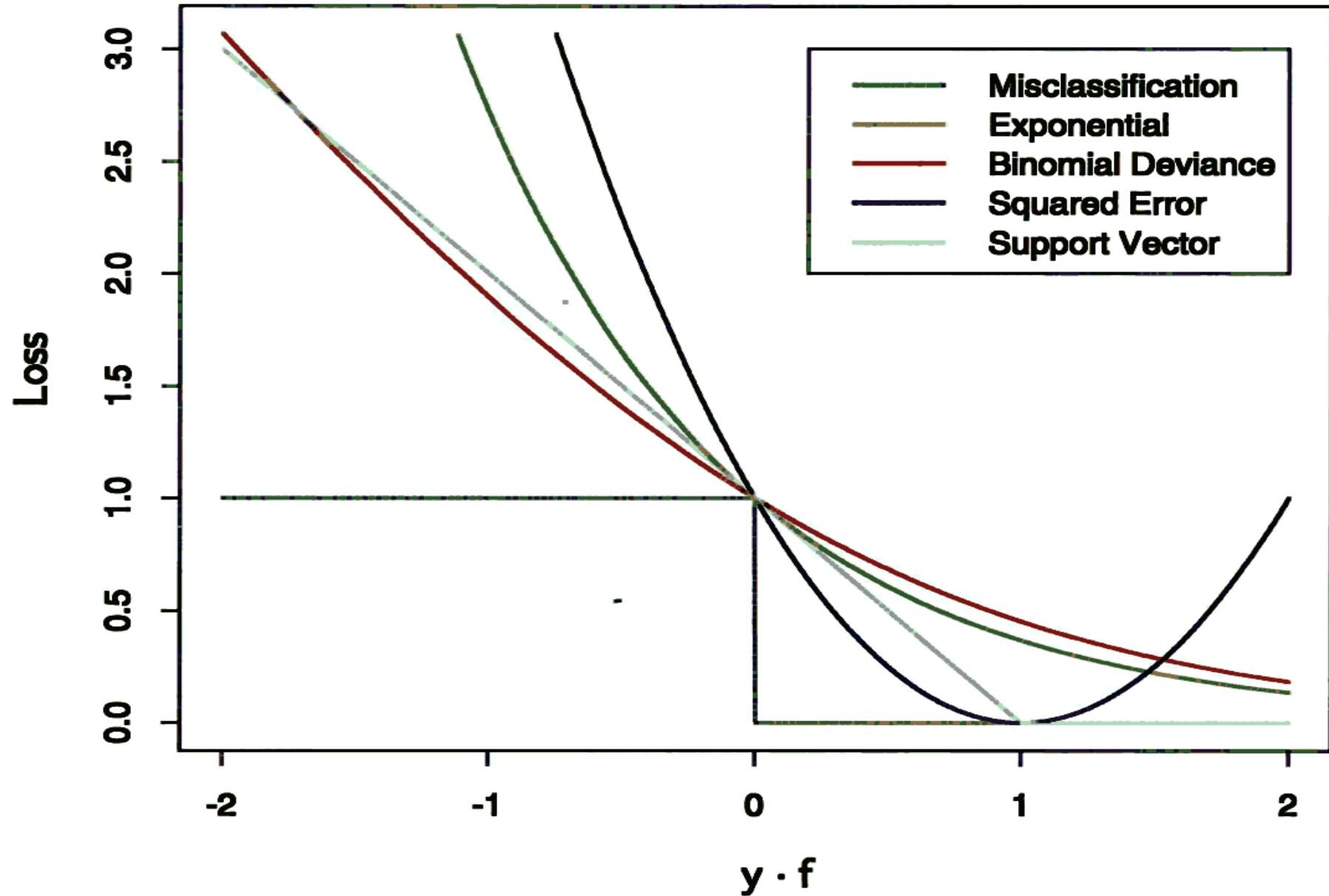
Learning in Games

- Learning through self-play.
- Backgammon (TD-Gammon).
- Samuel's checker program.

8 SUMMARY

- Important Loss Functions
- Learning Algorithm Characteristics Comparison
- More Learning
- Data Sets
- Journals
- Annual Conferences
- Recommended Literature

Important Loss Functions



Learning Algorithm Characteristics Comparison

Some characteristics of different learning methods. Learning methods.

Key: ● = good, ● = fair, and ● = poor.

Characteristic	Neural nets	SVM	Trees	MARS	k-NN, kernels
Natural handling of data of "mixed" type	●	●	●	●	●
Handling of missing values	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●
Computational scalability (large N)	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●
Interpretability	●	●	●	●	●
Predictive power	●	●	●	●	●

More Learning

- Concept Learning
- Bayesian Learning
- Computational Learning Theory (PAC learning)
- Genetic Algorithms
- Learning Sets of Rules
- Analytical Learning

Data Sets

- UCI Repository:
<http://www.ics.uci.edu/mlearn/MLRepository.html>
- UCI KDD Archive:
<http://kdd.ics.uci.edu/summary.data.application.html>
- Statlib:
<http://lib.stat.cmu.edu/>
- Delve:
<http://www.cs.utoronto.ca/delve/>

Journals

- Journal of Machine Learning Research
- Machine Learning
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Neural Computation
- Neural Networks
- IEEE Transactions on Neural Networks
- Annals of Statistics
- Journal of the American Statistical Association
- ...

Annual Conferences

- Algorithmic Learning Theory (ALT)
- Computational Learning Theory (COLT)
- Uncertainty in Artificial Intelligence (UAI)
- Neural Information Processing Systems (NIPS)
- European Conference on Machine Learning (ECML)
- International Conference on Machine Learning (ICML)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Artificial Neural Networks (ICANN)

Recommended Literature

- [Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [HTF01] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
- [Alp04] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.
- [Hut05] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005. <http://www.hutter1.net/ai/uaibook.htm>.