INTRODUCTION TO
STATISTICAL MACHINE LEARNING

Marcus Hutter
Canberra, ACT, 0200, Australia

Machine Learning Summer School
MLSS-2008, 2 – 15 March, Kioloa

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↔Unsupervised↔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

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 \( \text{income} > \theta_1 \) AND \( \text{savings} > \theta_2 \) 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

<table>
<thead>
<tr>
<th>scope of my lecture</th>
<th>scope of other lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>(machine) learning / statistical</td>
<td></td>
</tr>
<tr>
<td>induction ⇔ prediction</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td></td>
</tr>
<tr>
<td>independent identically distributed ⇔ sequential / non-iid</td>
<td></td>
</tr>
<tr>
<td>online learning ⇔ offline/batch learning</td>
<td></td>
</tr>
<tr>
<td>passive prediction ⇔ active learning</td>
<td></td>
</tr>
<tr>
<td>parametric ⇔ non-parametric</td>
<td></td>
</tr>
<tr>
<td>conceptual/mathematical ⇔ computational issues</td>
<td></td>
</tr>
<tr>
<td>exact/principled ⇔ heuristic</td>
<td></td>
</tr>
<tr>
<td>supervised learning ⇔ unsupervised ⇔ RL learning</td>
<td></td>
</tr>
</tbody>
</table>
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:** $\text{Even} = \{2, 4, 6\}$, $\text{Odd} = \{1, 3, 5\} \subseteq \Omega$
- **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:

- $P(\emptyset) = 0 \leq P(A) \leq 1 = P(\Omega)$,
- $P(A \cup B) + P(A \cap B) = P(A) + P(B)$,
- $P(A \cap B) = P(A|B)P(B)$. 
Probability Jargon

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

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: $P(\theta|1101) = \frac{P(1101|\theta)P(\theta)}{\sum_{\theta} P(1101|\theta)P(\theta)} \propto \theta^3(1 - \theta)$ (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: $\mathbb{E}[f|\ldots] = \sum_{\theta} f(\theta)P(\theta|\ldots)$, e.g. $\mathbb{E}[\theta|1101] = \frac{2}{3}$

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

Probability density: $P(\theta) = \frac{1}{\varepsilon}P([\theta, \theta + \varepsilon])$ for $\varepsilon \to 0$
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 \( \mathbf{x} := (1 \equiv x^{(0)}, x^{(1)}, \ldots, x^{(d)}) \in \mathbb{R}^{d+1} \)
- Real-valued noisy response \( y \in \mathbb{R} \).
- Linear regression model:
  \( \hat{y} = f_w(\mathbf{x}) = w_0 x^{(0)} + \ldots + w_d x^{(d)} \)
- Data: \( D = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \)
- Error or loss function:
  Example: Residual sum of squares:
  \( \text{Loss}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - f_w(\mathbf{x}_i))^2 \)
- 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 $\alpha \|w\|_2^2$.

**Lasso:** Add $\alpha \|w\|_1$.

**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: \( \mathcal{Y} = \{\text{spam,non-spam}\} \cong \{-1, 1\} \) (or \( \{0, 1\}\))

Reduction to regression: Regard \( y \in \mathbb{R} \Rightarrow \hat{\omega} \) from linear regression.

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

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

Log-odds:
\[
\log \frac{P(y=1|x,D)}{P(y=0|x,D)} := f_{\hat{\omega}}(x)
\]

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 $x \in \mathbb{R}^d$.

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

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

Examples:

- Linear regression: $p = d$ and $\phi_i(x) = x_i$.

- 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 ...
Linear Methods for Regression

Piecewise Constant

Piecewise Linear

$\xi_1$  $\xi_2$

Continuous Piecewise Linear

Piecewise-linear Basis Function

$(X - \xi_1)_+$
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 \( x \):

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

Nearest-Neighbor Kernel:

\( K(x, x_i) = 1 \) if \( \left| x - x_i \right| < a \) and 0 else

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

\( K(x, x_i) = \max\{0, a^2 - (x - x_i)^2\} \)
Regularization & 1D Smoothing Splines

to avoid overfitting if function class is large

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

- \( \lambda = 0 \)
  \[ \Rightarrow \hat{f} = \text{any function through data} \]

- \( \lambda = \infty \)
  \[ \Rightarrow \hat{f} = \text{least squares line fit} \]

- \( 0 < \lambda < \infty \)
  \[ \Rightarrow \hat{f} = \text{piecewise cubic with continuous derivative} \]
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:** \( f_{w,k}(\mathbf{x}) = \sigma(\sum_{j=0}^{M} w_{kj}^{(2)} z_j) \)

- **Sigmoidal activation functions:**
  \( h() \) and \( \sigma() \) \( \Rightarrow f \) non-linear

- **Goal:** Find network weights best modeling the data:

- **Back-propagation algorithm:**
  Minimize \( \sum_{i=1}^{n} ||y_i - f_{w}(x_i)||_2^2 \)
  w.r.t. \( w \) by gradient decent.
• 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 $\alpha$ minimizing $\sum_{i=1}^{n} L(y_i, (K\alpha)_i) + \lambda \alpha^\top K\alpha$.
- and Kernel $K_{ij} = K(x_i, x_j)$ following from $J$. 
Maximum Margin Classifier

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

- Linear boundary for \( \phi_b(x) = x^{(b)} \).
- Boundary is determined by Support Vectors (circled data points).
- Margin negative if classes not separable.
Sparse Kernel Methods / SVMs

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

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

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

depends only on $\phi$ 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 $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

• 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:
  $$\text{Err}_P(f) = \mathbb{E}_P[(y - f(x))].$$

• 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 $\lambda$,
- 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 $(a, \lambda, k, d)$ 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 $\lambda$, 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 $= \text{effective } \# \text{ of parameters to loss}.$
- 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 $\text{Rank}$ 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 $\Rightarrow$ $\text{Rank}$ 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 $\Rightarrow$ $\text{Rank}$ is small.

**LoRP:** Select model complexity $c$ that has minimal loss $\text{Rank}$

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:

\[
P(x_1)P(x_2)P(x_3)P(x_4|x_1 x_2 x_3)P(x_5|x_1 x_3)P(x_6|x_4)P(x_7|x_4 x_5)
\]
Additive Models & Trees & Related Methods

Generalized additive model:  \( f(x) = \alpha + f_1(x_1) + \ldots + f_d(x_d) \)

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 $\mu_1, \ldots, \mu_K$.
  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 $\hat{f}(x) = \frac{\sum_{i=1}^{n} K(x, x_i)y_i}{\sum_{i=1}^{n} K(x, x_i)}$. 
Approximate (Variational) Inference

Approximate full distribution $P(z)$ by $q(z)$

Popular: Factorized distribution:

$q(z) = q_1(z_1) \times ... \times q_M(z_M)$.

Measure of fit: Relative entropy

$KL(p||q) = \int p(z) \log \frac{p(z)}{q(z)} \, dz$

Red curves: Left minimizes $KL(P||q)$, Middle and Right are the two local minima of $KL(q||P)$. Examples: Gaussians
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:**
  \[
  \mathbb{E}[f] = \int f(z)p(z)dz \simeq \frac{1}{L} \sum_{l=1}^{L} f(z_l)p(z_l)/q(z_l),
  \]
  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 $q(z|z') = q(z'|z)$. Sample $z_{l+1}$ from $q(\cdot|z_l)$ but accept only with probability $\min\{1, p(z_{l+1})/p(z_l)\}$.

**Gibbs Sampling:** Metropolis with $q$ leaving $z$ unchanged from $l \sim l + 1$, except resample coordinate $i$ from $P(z_i|z\setminus_i)$.

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**: \( P(x|\theta, \pi) = \sum_k \pi_k P_k(x|\theta_k) \)

- **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 $\alpha_i$ 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 $f : \mathcal{X} \to \mathcal{Y}$ from I/O data $\{(x_i, y_i)\}$

**Unsupervised learning:**
Find pattern in data $(x_1, ..., x_n)$ 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 distance\(^2\) of data points from cluster centers:
  \[
  J(k, \mu) := \sum_{i=1}^{n} ||x_i - \mu_{k_i}||^2
  \]
- Choose centers \( \mu_k \) initially at random.
- **M-step**: Minimize \( J \) w.r.t. \( k \):
  Assign each point to closest center
- **E-step**: Minimize \( J \) w.r.t. \( \mu \):
  Let \( \mu_k \) 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:
\[ P(x | \pi \mu \Sigma) = \sum_{i=1}^{K} \text{Gauss}(x | \mu_k, \Sigma_k) \pi_k \]

Maximize likelihood \( P(x | ...) \) w.r.t. \( \pi, \mu, \Sigma \).

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

M-Step: Re-estimate \( \pi, \mu, \Sigma \) (take empirical mean/variance) given \( \gamma_{ik} \).
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: \( P(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|x_1 \ldots x_{i-1}) \)

Independent identically distributed (roulette, dice, classification):
\( P(x_1 \ldots x_n) = P(x_1)P(x_2) \ldots P(x_n) \)

First order Markov chain (Backgammon):
\( P(x_1 \ldots x_n) = P(x_1)P(x_2|x_1)P(x_3|x_2) \ldots P(x_n|x_{n-1}) \)

Second order Markov chain (mechanical systems):
\( P(x_1 \ldots x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1x_2) \times \ldots \times P(x_n|x_{n-1}x_{n-2}) \)

Hidden Markov model (speech recognition):
\( P(x_1 \ldots x_n) = \int P(z_1)P(z_2|z_1) \ldots P(z_n|z_{n-1}) \times \prod_{i=1}^{n} P(x_i|z_i) d\mathbf{z} \)
Sequential Decision Theory

Setup:

For \( t = 1, 2, 3, 4, \ldots \)

Given sequence \( x_1, x_2, \ldots, x_{t-1} \)

(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

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

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

\[
\begin{array}{c|cc}
\text{Loss} & \text{sunny} & \text{rainy} \\
\hline
\text{umbrella} & 0.1 & 0.3 \\
\text{sunglasses} & 0.0 & 1.0 \\
\end{array}
\]

Goal: Minimize expected Loss: 

\[ \hat{y}_t = \arg \min_{y_t} \mathbb{E}[\text{Loss}(x_t, y_t)|x_1 \ldots x_{t-1}] \]

Greedy minimization of expected loss is optimal if:

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

Examples:

\[ \hat{y} = \text{mean} / \text{median} / \text{mode} \quad \text{of} \quad P(x_t|\cdots) \]

\[ \text{Loss} = \text{square} / \text{absolute} / \text{0-1 error function} \]
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

Performance standard

Agent

Critic

Sensors

feedback

learning goals

Environment

Learning element

changes

knowledge

Problem generator

Performance element

Actuators

Problem generator

Learning element

Performance standard
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 though 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

- **Misclassification**
- **Exponential**
- **Binomial Deviance**
- **Squared Error**
- **Support Vector**
### Learning Algorithm Characteristics Comparison

Some characteristics of different learning methods.\textit{\noindent Key:} ● = good, ○ = fair, and ■ = poor.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Neural nets</th>
<th>SVM</th>
<th>Trees</th>
<th>MARS</th>
<th>k-NN, kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural handling of data of &quot;mixed&quot; type</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Handling of missing values</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Robustness to outliers in input space</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Insensitive to monotone transformations of inputs</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Computational scalability (large $N$)</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Ability to deal with irrelevant inputs</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Ability to extract linear combinations of features</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Interpretability</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Predictive power</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
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:

- 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


