A primer on Machine Learning

Paolo Marcatili
CBS, DTU
paolo.marcatili@gmail.com
Schedule

9.00 to 10.15
Overview

10.30 to 12.00
Examples

13.00 to 17.00
Practicals
Lesson goals

- See what’s out there
- Be able to understand the main pitfalls
- Train your first predictor
Agenda

Overview

Assessments

Supervised Learning
  SVM
  HMM
  Bayesian Networks
  Neural Networks
  Decision Trees – RF

Unsupervised Learning
  Clustering

Caveats
"Field of study that gives computers the ability to learn without being explicitly programmed"
1959, A. Samuel

Artificial Intelligence
Theoretical
Idealistic
Models
What is Machine learning *today*

A way to deal with
- large
- noisy
- heterogeneous data

Artificial Intelligence
Theoretical
Idealistic
Models

Statistics
Practical
Pragmatic
Black-boxes
Why

- Large amount of data
- Large dimensionality
- Complex dynamics
- Data Noisiness
Why

Large amount of data
Large dimensionality
Complex dynamics
Data Noisiness
Computational efficiency
Because we can
How

Numerical analysis
Graphs
Systems theory
Geometry
Statistics
Probability
How

Numerical analysis

Graphs

Systems theory

Geometry

Statistics

Probability

Probability and statistics are fundamental. They provide a solid framework for creating models and acquire **knowledge**
Datasets

Most common data used with ML:

Genomes (genes, promoters, phylogeny, regulation...)

Proteomes (secondary/tertiary structure, disorder, motifs, epitopes...)

Clinical Data (drug evaluation, medical protocols, tool design...)

Interactomic (PPI prediction and filtering, complexes...)

Metabolomic (metabolic pathways identification, flux analysis, essentiality)
Methods

Machine Learning can
Methods

Machine Learning can

Predict unknown function values
Methods

Machine Learning can

- Predict unknown function values
- Infer classes and assign samples
Methods

Machine Learning can

- Predict unknown function values
- Infer classes and assign samples
Methods

Machine Learning can not
Methods

Machine Learning can not

Provide knowledge
Methods

Machine Learning cannot provide knowledge.
Methods

Machine Learning can not

Provide knowledge

Learn
Machine learning

part I: the training

Input Variables
(e.g. peptide sequence)

Data to be “learned”
(e.g. aggregation propensity)
Machine learning

part I: the training

Input Variables
(e.g. peptide sequence)

Data to be “learned”
(e.g. aggregation propensity)

prediction

(aggregation propensity)
Machine learning

part I: the training

Input Variables (e.g. peptide sequence)

Data to be “learned” (e.g. aggregation propensity)

if prediction ≠ training data

change something (method, variables, project)

prediction (aggregation propensity)
Machine learning

part I: the training

Input Variables (e.g. peptide sequence)

Data to be “learned” (e.g. aggregation propensity)

If prediction ≠ training data
change something (method, variables, project)

This is called overfitting

Prediction (aggregation propensity)
Machine learning

part II: the validation

NEW Input Variables (e.g. peptide sequence)

Measure the accuracy by comparing prediction with validation data
Agenda

Overview

Assessments

Supervised Learning
  SVM
  HMM
  Bayesian Networks
  Neural Networks
  Decision Trees – RF

Unsupervised Learning
  Clustering

Caveats
Validation Methods
Validation Methods

Split Data in 2
Not reliable nor efficient
Validation Methods

Training set

Validate on 1 sample

Repeat on each sample

Leave-one-out

Not efficient
Validation Methods

Train on (K-1) subsets
Validate on 1 subset
Repeat K time

Cross-fold validation
The most common
Caveats

Cross-fold is necessary

but *not* sufficient

Overfitting can be hidden:

Redundancy
Negative dataset
Overparametrization
Assessment

**True Positive Rate:** \( \frac{TP}{TP + FN} \)
Given the disease is present, the likelihood of testing positive.

**False Positive Rate:** \( \frac{FP}{TN + FP} \)
Given the disease is not present, the likelihood of testing positive.

**Positive Predictive Value:** \( \frac{TP}{TP + FP} \)
Given a test is positive, the likelihood disease is present
Assessment

receiver operating characteristic (ROC) is a graphical plot of the sensitivity vs. (1 - specificity) for a binary classifier system as its discrimination threshold is varied.

**True Positive Rate:** $\frac{TP}{TP + FN}$
Given the disease is present, the likelihood of testing positive.

**False Positive Rate:** $\frac{FP}{TN + FP}$
Given the disease is not present, the likelihood of testing positive.

**Positive Predictive Value:** $\frac{TP}{TP + FP}$
Given a test is positive, the likelihood disease is present.
ROC curve

Score to separate Positives from Negatives
High threshold -> less FP, more FN
Low threshold -> more FP, less FN
ML for large noisy heterogeneous data

Don’t use the same data to train and to assess. Sometimes it’s evident, sometimes it’s not

Use cross-fold validation

Data redundancy can be an issue

Don’t let your negative dataset be too negative
Agenda

Overview

Assessments

**Supervised Learning**
- SVM
- HMM
- Bayesian Networks
- Neural Networks
- Decision Trees – RF

**Unsupervised Learning**
- Clustering

Caveats
Supervised Learning

Basic Idea: use data+classification of known samples
find “fingerprints” of classes in the data
Supervised Learning

Basic Idea: use data+classification of known samples
find “fingerprints” of classes in the data

Example:
use microarray data, different condition
classes: genes related/unrelated to different cancer types
Support Vector Machines

Basic idea:
Plot your data in an N-dimensional space

Find the best hyperplane that separates the different classes

Further samples can be classified using the region of the space they belong to
Support Vector Machines

- Fail
- Pass

(weight vs. length)
Support Vector Machines

- **Fail**
- **Pass**

![Graph showing data points labeled as Fail and Pass. The graph includes a margin between the two classes.]
Support Vector Machines

Optimal Hyperplane (OHP)

simple kind of SVM (called an LSVM)

Support vectors
Support Vector Machines

What if data are not linearly separable?
Support Vector Machines

What if data are not linearly separable?

Allow mismatches

soft margins

(add a weight matrix)
What if data are not linearly separable?
Support Vector Machines

What if data are not linearly separable? The Kernel trick!

Only Inner product is needed to calculate Dual problem and decision function

Kernelization
Hidden Markov Models

There is a regular and a biased coin.

You don't know which one is being used.

During the game the coins are exchanged with a certain fixed probability

All you know is the output sequence
Hidden Markov Models

There is a regular and a biased coin.

You don't know which one is being used.

During the game the coins are exchanged with a certain fixed probability

All you know is the output sequence

HHTHTHTHTHTTTTTHHTHHHHHHHHHHTHTHTHHTHTHHHHTHTH

Given a set the parameters, which is the probability of the output seq.?  
Which parameters are more likely to have produced the output?  
Which coin was being used at a certain point of the sequence?
Hidden Markov Models
a) HIV is affecting 0.01% of population.
b) The HIV test, when performed on patients, is correct 99.9% of times.
b) The HIV test, when performed on uninfected people, is correct 99.99% of times.

If a person has a positive test, how likely is it for him to be infected?
Bayes Theorem

a) HIV is affecting 1 person over 10,000

b) The HIV test,
   on patients, is correct 999 times every 1000.
   on uninfected people, is correct 9.999 times every 10,000.
Bayes Theorem

a) HIV is affecting 1 person over 10,000
b) The HIV test,
on patients, is correct 999 times every 1000.
on uninfected people, is correct 9.999 times every 10,000.
Bayes Theorem

a) HIV is affecting 1 person over 10.000
b) The HIV test,
on patients, is correct 999 times every 1000.
on uninfected people, is correct 9.999 times every 10.000.
Bayes Theorem

\[ P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \]

a) HIV is affecting 0.01% of population.
b) The HIV test, when performed on patients, is correct 99.9% of times.
b) The HIV test, when performed on uninfected people, is correct 99.99% of times.

If a person has a positive test, how likely is it for him to be infected?
Bayes Theorem

\[ P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) P(A)}{P(B)} . \]

a) HIV is affecting 0.01% of population.
b) The HIV test, when performed on patients, is correct 99.9% of times.
b) The HIV test, when performed on uninfected people, is correct 99.99% of times.

If a person has a positive test, how likely is it for him to be infected?

\[ P(H|T) = \frac{P(T|H) P(H)}{P(T|H) P(H) + P(T|\neg H) P(\neg H)} \]

\[ P(H|T) = 49.97\% \]
Bayesian Networks

The probabilistic approach is extremely powerful but requires a huge amount of information/data for a complete representation.

Not all correlations or cause-effect relationships between variables are significative.
Bayesian Networks

The rationale:

You want to estimate the probability of hidden variables (e.g. cancer) By observing its effects (visible variables).

To do that you use Bayes theorem to pass from

\[ P(V|H) \]

To

\[ P(H|V) \]
Bayesian Networks

The probabilistic approach is extremely powerful but requires a huge amount of information/data for a complete representation.

Not all correlations or cause-effect relationships between variables are significative.

Consider only meaningful links!
Bayesian Networks

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?
- Variables: *Burglary, Earthquake, Alarm, JohnCalls, MaryCalls*
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call
Bayesian Networks

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?
- Variables: Burglary, Earthquake, Alarm, JohnCalls, MaryCalls
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

Bayes Theorem again!
I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

Variables: *Burglary, Earthquake, Alarm, JohnCalls, MaryCalls*

Network topology reflects "causal" knowledge:
- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Bayes Theorem again!
Bayesian Networks

Long story, but it gets easier because of some tricks:
B/E and J/M are only dependent through A
You can eliminate a lot of variables

Bayes Theorem again!
Neural Networks

Neural Networks interpolate functions
They have nothing to do with brains
Neural Networks interpolate functions
They have nothing to do with brains
Neural Networks

Neural Networks interpolate functions
They have nothing to do with brains
Neural Networks

Parameter settings: we have to learn weights from training data.

The typical algorithm (back-propagation) starts with random weights and then iteratively adjusts the weights.
Neural Networks

Parameter settings: avoid overfitting
Questions
Decision trees

Mimics the behavior of an expert
Decision trees

The idea is to use the variables to divide the sets into subsets that are more homogenous.
Decision trees

The idea is to use the variables to divide the sets into subsets that are more homogenous

Initial set: 100 play, 80 non play

If humid: 60 play, 80 non play
If non humid: 40 play, 20 non play
Decision trees

The idea is to use the variables to divide the sets into subsets that are more homogenous

Initial set: 100 play, 80 non play

If sunny: 30 play, 10 non play
If overcast: 30 play, 5 non play
If rainy: 40 play, 65 non play
Pros:  
Easy to interpret  
Statistical analysis  
Informative results

Cons:  
A single variable  
Not optimal  
Not robust

Decision trees

Majority rules!
Random Forests

Split the data in several subsets, construct a DT for each set

Each DT expresses a vote, the majority wins

Much more accurate and robust (bootstrap)
Random Forest

Training data

1) Generate bootstrap samples
Random Forest

Take out n samples and replace with duplicates
Usually n~ 1/3 of your samples

1) Generate bootstrap samples

Training data → training bag
Random Forest

1) Generate bootstrap samples

2) Select the best variable among $m$ randomly chosen to start the tree ($m$ is user submitted, can be tuned)
Random Forest

1) Generate bootstrap samples
2) Random variable selection
3) Fit unpruned decision trees
4) Apply to testing data & combine predictions
Random Forest

1) Generate bootstrap samples
   →
2) Random variable selection
   →
3) Fit unpruned decision trees
   →
4) Apply to testing data & combine predictions

Training data

Testing data
<table>
<thead>
<tr>
<th>Model</th>
<th>Non numeric data</th>
<th>Overfitting/overtraining</th>
<th>Missing data</th>
<th>Parametrization</th>
<th>Comput. time</th>
<th>Black box</th>
<th>Predictive power</th>
<th>Priors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Trees</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Easy</td>
<td>Fast</td>
<td>White</td>
<td>low</td>
<td>optional</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Easy</td>
<td>Medium</td>
<td>Grey</td>
<td>high</td>
<td>optional</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Medium</td>
<td>Slow</td>
<td>Black</td>
<td>Medium/high</td>
<td>no</td>
</tr>
<tr>
<td>SVM</td>
<td>No</td>
<td>sometimes</td>
<td>Sort</td>
<td>Medium</td>
<td>Fast</td>
<td>Black</td>
<td>high</td>
<td>no</td>
</tr>
<tr>
<td>HMMs</td>
<td>Yes</td>
<td>No</td>
<td>Sort</td>
<td>Easy</td>
<td>Fast</td>
<td>Grey</td>
<td>medium</td>
<td>optional</td>
</tr>
<tr>
<td>Bayesian networks</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Hard</td>
<td>Slow</td>
<td>White</td>
<td>high</td>
<td>recommended</td>
</tr>
</tbody>
</table>
Agenda

Overview

Assessments

Supervised Learning
   SVM
   HMM
   Bayesian Networks
   Neural Networks
   Decision Trees – RF

Unsupervised Learning
   Clustering

Caveats
Unsupervised Learning

If we have no idea of actual data classification, we can try to guess
Clustering

Put together similar objects to define classes
Clustering

Put together similar objects to define classes
Clustering

Put together similar objects to define classes

K-means
Hierarchical top-down
Hierarchical down-up
Fuzzy

How?
Clustering

Put together similar objects to define classes

How? Which metric?

Euclidean
Correlation
Spearman Rank
Manhattan
Clustering

Put together similar objects to define classes

How?  Which metric?  Which “shape”?  Compact  Concave  Outliers  Inner radius  cluster separation
Hierarchical Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method
Hierarchical Clustering

• We start with every data point in a separate cluster
• We keep merging the most similar pairs of data points/clusters until we have one big cluster left
• This is called a bottom-up or agglomerative method
Hierarchical Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method
Hierarchical Clustering

• We start with every data point in a separate cluster
• We keep merging the most similar pairs of data points/clusters until we have one big cluster left
• This is called a bottom-up or agglomerative method
Hierarchical Clustering

- We start with every data point in a separate cluster.
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left.
- This is called a bottom-up or agglomerative method.
Hierarchical Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method
Hierarchical Clustering

- We start with every data point in a separate cluster.
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left.
- This is called a bottom-up or agglomerative method.
Hierarchical Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers

- Assign each sample to the closest center

- Recompute centers (samples average)

- Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
**K-means**

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers

- Assign each sample to the closest center

- Recompute centers (samples average)

- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
K-means

- Start with K random centers

- Assign each sample to the closest center

- Recompute centers (samples average)

- Repeat until converged
K-means

- Start with K random centers

- Assign each sample to the closest center

- Recompute centers (samples average)

- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers
- Assign each sample to the closest center
- Recompute centers (samples average)
- Repeat until converged
K-means

- Start with K random centers

- Assign each sample to the closest center

- Recompute centers (samples average)

- Repeat until converged
K-means

• Start with K random centers

• Assign each sample to the closest center

• Recompute centers (samples average)

• Repeat until converged
Agenda

Overview

Assessments

Supervised Learning
  SVM
  HMM
  Bayesian Networks
  Neural Networks
  Decision Trees – RF

Unsupervised Learning
  Clustering

Caveats
Data independence

Training set, Test set and Validation set must be clearly separated

E.g. neural network to infer gene function from sequence

training set: annotated gene sequences, deposit date before Jan 2007
test set: annotated gene sequences, deposit date after Jan 2007

But annotation of new sequences is often inferred from old sequences!
Biases

Data should be unbiased, i.e. it should be a good sample of our “space”

E.g. neural network to find disordered regions
training set: solved structures, residues in SEQRES but not in ATOM

But solved structures are typically small, globular, cytoplasmatic proteins
Take-home message

Always look at data. ML methods are extremely error-prone

Use probability and statistics where possible

Be careful with biases, redundancy, hidden variables

Be careful with overfitting and overparametrizing
References


• http://see.stanford.edu/see/courseinfo.aspx?coll=348ca38a-3a6d-4052-937d-cb017338d7b1
Questions
Occam’s razor

Don’t use models that are too complex for your data
Occam’s razor

Don’t use models that are too complex for your data
Occam’s razor

Don’t use models that are too complex for your data

Rule of thumb: at least 3 samples for 1 parameter