Overview

Large datasets in science and technology

Practical Uses:

a) Business Transactions
b) Digging Social Media
c) AI/Robotics

Physics / Astro / Bio :

a) HEP: CERN LHC $\rightarrow$ 25 GB/s (with all jets)
   Nominal rate 69 TB/s $= 6 \times 10^8$ events/s $\rightarrow$ 100 events/s
   16 GB/s $\rightarrow$ 1 TB/day
b) Astro: Large survey telescopes
   LSST $\rightarrow$ 20 TB/night

c) Bio: Large genome centers like
   New York Genome Center $\sim$ 10 TB/day
Need for automated analysis to flag interesting patterns. \( \rightarrow \) Machine Learning (ML)

The book we follow emphasizes probabilistic approach. Not all methods are dependent on probabilistic formulation but we will mostly take prob models as these are in familiar territory for physicists.

ML and Statistics overlaps a lot. We will quickly cover relevant corners of statistics.

**Types of machine learning**

a) **Supervised Learning**

Given \( D = \{ (x_i, y_i) \} \) \( i = 1 \cdots n \) \( \rightarrow \) training set

- Feature vectors \( x_i \in \mathbb{R}^d \)
- Outputs categorical, real or ordinal

Predict for \( x_i \), \( y = h(x) \)
b) **Unsupervised Learning**

Only $\mathbb{X} \subseteq \mathbb{F}^n$ are given. Find hidden structure.

Examples: clustering, dimension reduction, ...

c) **Reinforcement Learning**

Depending on performance in some task, reward or punishment signal is sent to the algorithm. The algorithm then tweaks itself to perform better.

Dynamic training data.

Examples: Game playing systems
Supervised Learning

Regression:

Linear Regression

Non-linear Regression

How complex should our fitting function be?

Multivariate examples:
Classification

\{ (x, y) \} \quad \text{y \in \{0, 1\}, binary classification}

Given any \( x \), give us a probability \( y \) corresponding \( y \) being 0, 1.

Generalization to \( y \in \{1, \ldots, C\} \), multiclass classification

Example 1: Given a galaxy image, classify it as elliptical, spiral, ...

2) Given some protein sequence, assign it to a structural or functional class.
Probabilistic approach to supervised learning:
Wanted a function approx.
\[ \hat{y} = f(x) \]
Construct \( p(y|x,D) \)
\[ \hat{y} = \text{argmax}_y p(y|x,D) = f(x) \]
Maximum a posteriori (MAP) estimate.

Practical applications:
1) document classification, e.g. spam filtering
2) Image classification, e.g. handwriting recognition,
3) Object recognition, e.g. face detection, ..
Unsupervised Learning

Knowledge discovery, density estimation,
Want: \( P(x; \theta) \) as opposed to \( P(y; x; \theta) \)

Discovering clusters

Discovering latent factors

\[ x_i = A z_i = \sum_{k=1}^{k} a_k z_{k_i} \]

\[ A = [a_1 \ldots a_k] \]
Examples: Principal component analysis
\[ \alpha_i \text{'s orthogonal (PCA)} \]

Independent Component Analysis (ICA)
\[ z_i \text{'s are independent} \]

Additional examples:

a) Discovering underlying graph structures

Gene expression
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
\[ \Rightarrow \]
Some basic concepts

Parametric vs non-parametric models

The number of parameters fixed

Parametric models for regression and classification

Regression
\[ y(x) = \mathbf{w}^T x + \epsilon \]

residual error

\[ p(y | x, \sigma) = \mathcal{N}(y | \mathbf{w}(x), \sigma^2(x)) \]

more generally \( \mathbf{w}(x) = \mathbf{w}^T \phi(x) \)

For example
\[ \phi(x) = [1, x, x^2, \ldots, x^n] \]

\[ y_i = w_0 + w_1 x_i + w_2 x_i^2 + \cdots \]
Classification

Logistic regression

\[ P(y|x, w) = \text{Ber}(y | \mu(x)) \]

\[ \mu(x) = \sigma \text{igm}(w^T x) \]

\[ \sigma(\eta) = \frac{1}{1 + e^{-\eta}} \]

Decision rule

\[ \hat{y}(x) = \begin{cases} 1 & P(y|x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \]

Linearly separable.
Advantage: Efficient
Disadvantage: Less flexible

\[ 0 \quad 0 \quad 6 \]
\[ x \quad 1 \quad 6 \quad 0 \quad 6 \]
\[ x \quad x \quad 6 \quad 0 \quad 6 \]
\[ x \quad x \quad x \quad x \quad 6 \]

Not linearly separable.
Need more complex parametrization.

Non-parametric classifier: Example
K-nearest neighbor model.

\[
\hat{p}(y = c | x, D, K) = \frac{1}{K} \sum_{i \in N_k(x, D)} \mathbb{I}(y_i = c)
\]
If the distribution $p(x)$ is 'regular', and the number of data points $N$ go to infinity, $k$-NN would actually give good results, since it estimates local density well.

**Trouble: Curse of dimensionality**

Example: data distributed uniformly in a hypercube. A smaller cube of size $s$ has fractional volume $s^D$.

The number of points within the smaller volume $K = Ns^D$, implying $s = (K/N)^{1/D} = f^{1/D}$.

For large $D$ and practical values of $K, N$, $s$ is often close to 1. For example, if $K \sim 10$, $D \sim 10$, $N \sim 10^4$,

$\Rightarrow \ s \sim 5$

So points inside are not close by.
Overfitting:
We usually minimize
\[ \text{err}(f, D) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x_i), y_i) \]

Regression
Curve fitting

Decision Boundary
Classification

We do not want
\[ f(x_1, x_2) = x_1 \]
or
\[ f(x_1, x_2) = x_2 \]

even if it does a perfect job on training data.
Model Selection
- Too simple a model $\rightarrow$ increased error
- Too complex a model $\rightarrow$ ?
  - training
  - reduces error on test set.

Generalization error
- error on 'future' data

Hands on MATLAB exercise

U shaped curve

Common strategy to guard
against overfitting!

Training data \rightarrow \text{Validation set (test set)}

Often.training set 80\%
validation set 20\%

If the dataset is small, often not enough statistics.

Popular solution: Cross validation (CV)
Split data \text{\& fold}

Validate \rightarrow train

Validate \rightarrow train

etc.
Popular choice $K = 5$

5-fold CV has $80\%$, $20\%$ split, but many splits.

For size $N$ training set

$K = N \Rightarrow$ Leave one out CV

LOOCV

Choosing number of nearest neighbors $K_{NN}$.

\[ \text{Parameter in regression}\]

No free lunch theorem:

(Wolpert 1996)

Imagine I give you classification.
task: \( D = \{(x_i, y_i)\}_{i=1}^{\infty} \) is the training set

\( D^+ = \{(x_{j+N}, y_{j+N})\}_{j=1}^{M} \) is the validation set

For simplicity assume no \( x \)'s are the same

Let \( f : X \rightarrow Y \)

On \( \{x_1, \ldots, x_{N+M}\} \) there are \( 2^{N+M} \) choices for \( f \). Consider a prob. distrn \( P(f) \) that uniformly samples from these \( 2^{N+M} \) functions.

If \( D^+ \) and \( D^+ \) are generated by \( f \) and algorithm \( A \) training on \( D \) produces a function \( \hat{f} \)

\[
\text{Generalization error } \text{err}(\hat{f}, D^+) = \frac{1}{M} \sum_{j=1}^{M} I(\hat{f}(x_{j+N}) \neq y_{j+N})
\]

\[
\mathbb{E}[\text{err}(\hat{f}, D^+)] = \mathbb{E}[\text{err}(\hat{f}, D^+)] P(f) = \frac{1}{2}
\]

If algorithm \( A \rightarrow f_A, D^+ \) and \( B \rightarrow f_B, D^+ \)

Their average performance over the validation set is the same
So it cannot be that one algorithm always outperforms another under all circumstances.

> Importance of tailoring algorithms to match the structure of data to be processed.

Role of bias in learning.