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 → 25 GB/s (with all cuts)
Nominal rate 68 TB/s ≡ 6×10^8 events/s → 100 event/s
1 GB/s → 100 TB/day

b) Astro: Large survey telescopes
LSST → 20 TB/night

c) Bio: Large genome centers like New York Genome Center ~ 10 TB/day.

Need for automated analysis to flag interesting patterns. → 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 $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ → training set

x_i: feature vectors, $x_i \in \mathbb{R}^D$

y_i: outputs, categorical, real or ordinal

Predict for x. $y = h(x)$

b) Unsupervised Learning

Only $\{x_i\}_{i=1}^{N}$ are given. Find hidden structure.

Examples: clustering, dimension reduction, ..

c) Reinforcement Learning

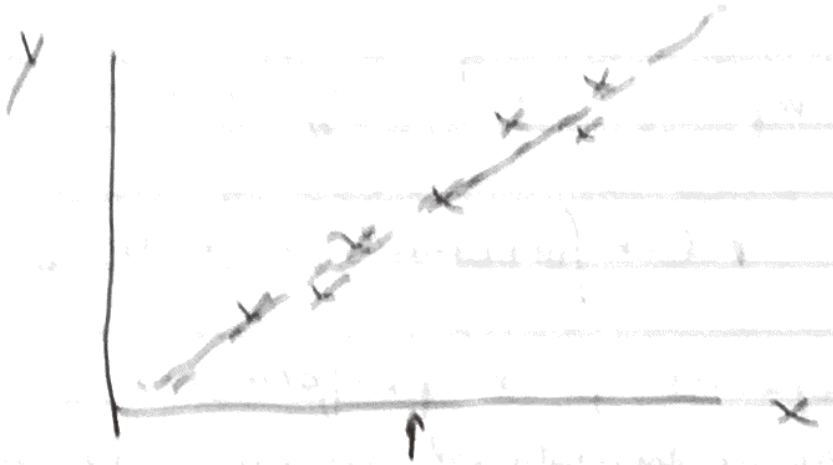
Depending on performance in some task reward or punishment signal is sent to the algorithm. The algorithm then tweaks itself to perform better.

Dynamic training data.

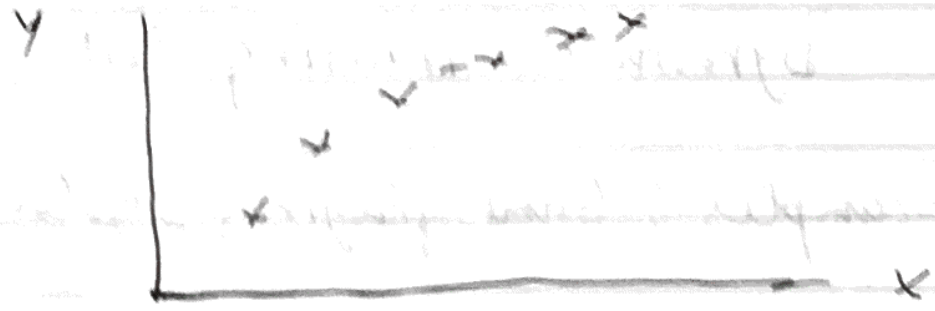
Examples: Game playing systems.

Supervised Learning

Regression:



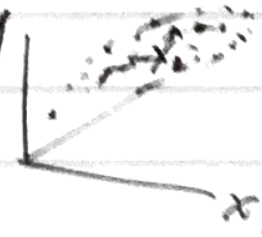
Linear regression



Non linear regression.

How complex should our fitting function be?

Multivariate examples:



Classification

$\{(x_i, y_i)\}$ $y_i \in \{0, 1\}$, binary classification

Given any x, give us a probability of corresponding y being 0, 1.

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

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

b) Given some protein sequence, assign it to a structural or functional class.

Probabilistic approach to Supervised learning:

Want a function approx.

$\hat{y} = \hat{f}(x)$

Construct $p(y|x, \mathcal{D})$

$$\hat{y} = \underset{y}{\operatorname{argmax}}\ p(y|x, \mathcal{D}) = \hat{f}(x)$$

Maximum a posteriori (MAP) estimate.

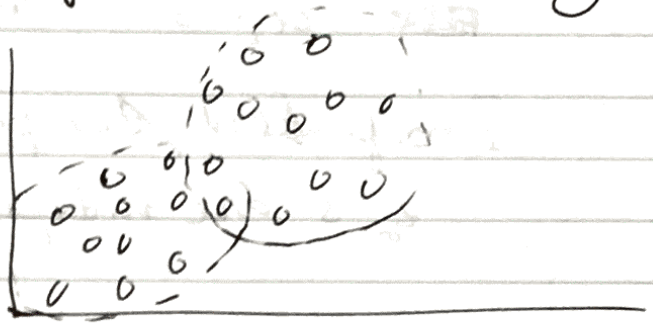
Practical applications:

a) document classification, e.g. spam filtering

b) Image classification, e.g. handwriting recognition,

c) Object recognition, e.g face detection, ...

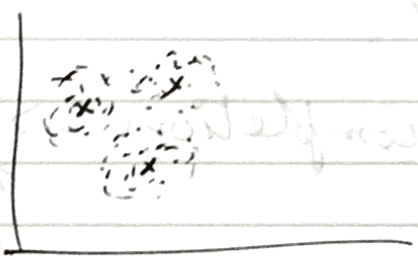
Unsupervised Learning



Knowledge discovery, density estimation, ..

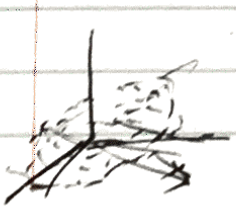
Want. $p(x_i|\theta)$ as opposed to $p(y_i|x_i;\theta)$

Discovering clusters



Discovering Latent factors

$$x_i = A z_i = \sum_{k=1}^{k} a_k z_{ki} \qquad A = [a_1 \cdots a_k]$$



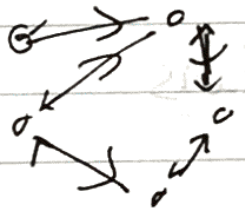
Examples: Principal component analysis (PCA)
a_i's orthogonal

Independent component analysis (ICA)
z_i's are independent.

Additional examples:

a) Discovering underlying graph structures

Gene expression ⇒ Networks



b) Matrix completion → imputing missing data,
recommendation system.

Some basic Concepts

Parametric vs non-parametric models

The number of parameters fixed

Parametric models for regression and classification

Regression

$$y(x) = w^T x + \epsilon$$

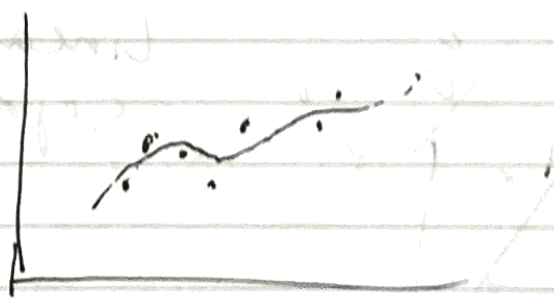
ϵ ↑ residual error

$$p(y \mid x, \theta) = \mathcal{N}(y \mid \mu(x), \sigma^2(x))$$

$$\mu(x) = w^T x$$

More generally $\mu(x) = w^T \phi(x)$

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



$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \cdots$$

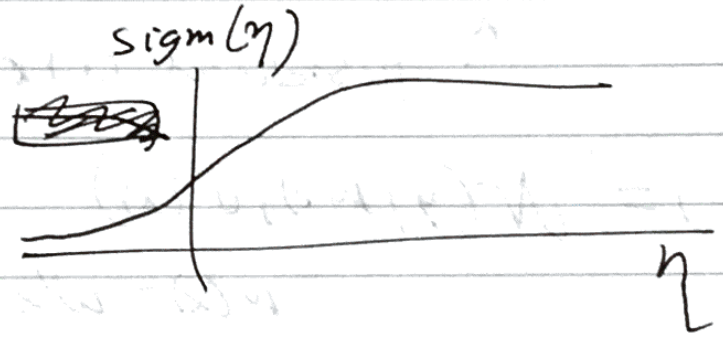
Classification

Logistic regression

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

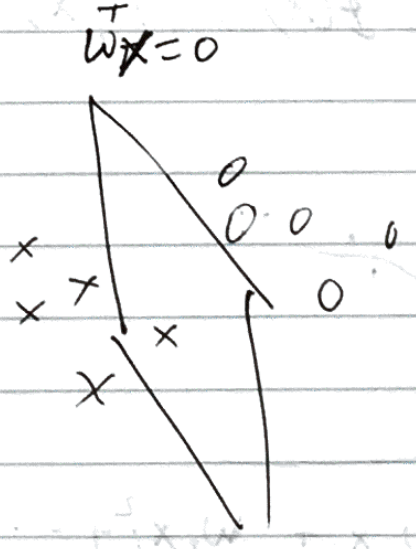
$$\mu(x) = \text{sigm}(w^T x)$$

$$\text{sigm}(\eta) = \frac{1}{1+e^{-\eta}}$$



Decision rule $\hat{y}(x) = 1 \quad P(y|x) > \frac{1}{2}$

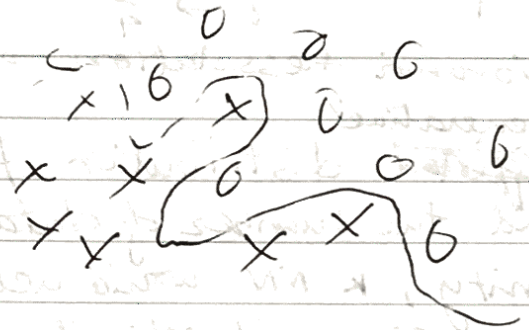
$= 0 \quad \text{otherwise}$



linearly separable.

Advantage: Efficient

Disadvantage: Less flexible



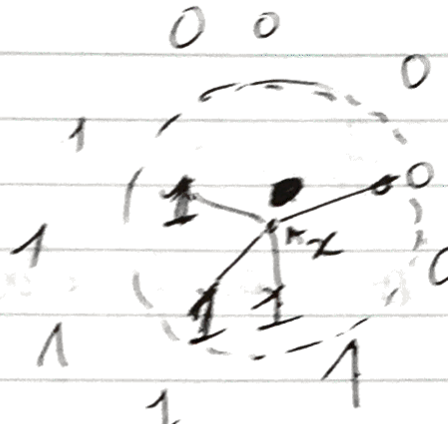
not linearly separable.

Need more complex parametrization

Non-parametric Classifier: Example

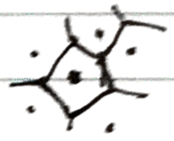
K-nearest neighbor model.

$$p(y=c \mid x, D, K) = \frac{1}{K} \sum_{i \in N_K(x,D)} I(y_i = c)$$



$1 \to \frac{3}{4}$

$0 \to \frac{1}{4}$

$K=1 \to$ Voronoi tesselation $\to$ 

If the generative 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 unit hypercube

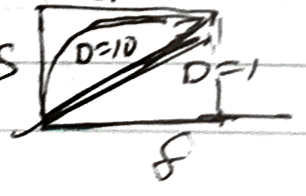
A smaller cube of size s has fractional volume s^D

The number of points within the smaller volume $K \approx N s^D$, implying $s \approx \left(\frac{K}{N}\right)^{1/D} = f^{1/D}$

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 involved are not closeby.

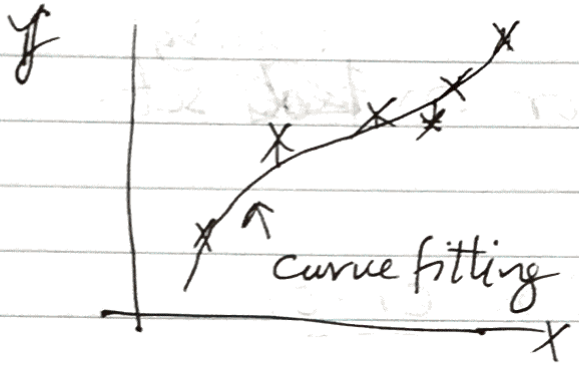
Overfitting:

We usually minimize

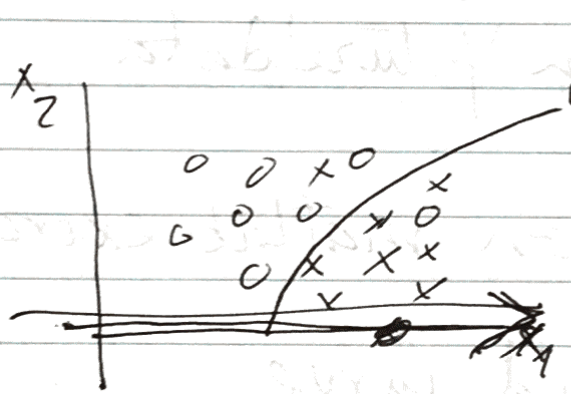
$$err(f, D) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)$$

← least squares

← misclassification count.

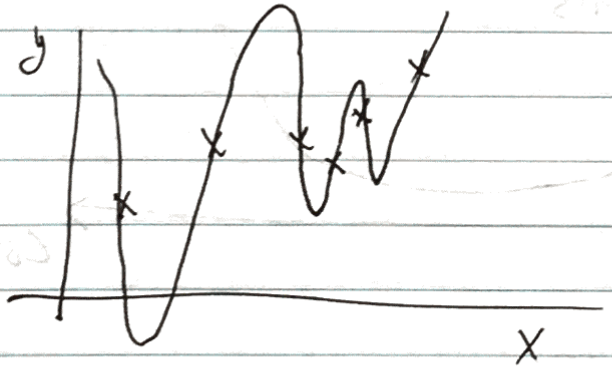


Regression

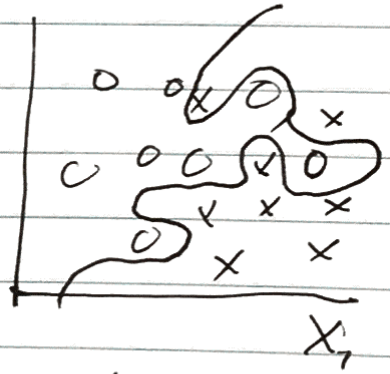


Classification

We do not want



OR



even if it does a perfect job on training data.

Model Selection

Too simple a model ⟹ increased error

Too complex a model ⟹ ?

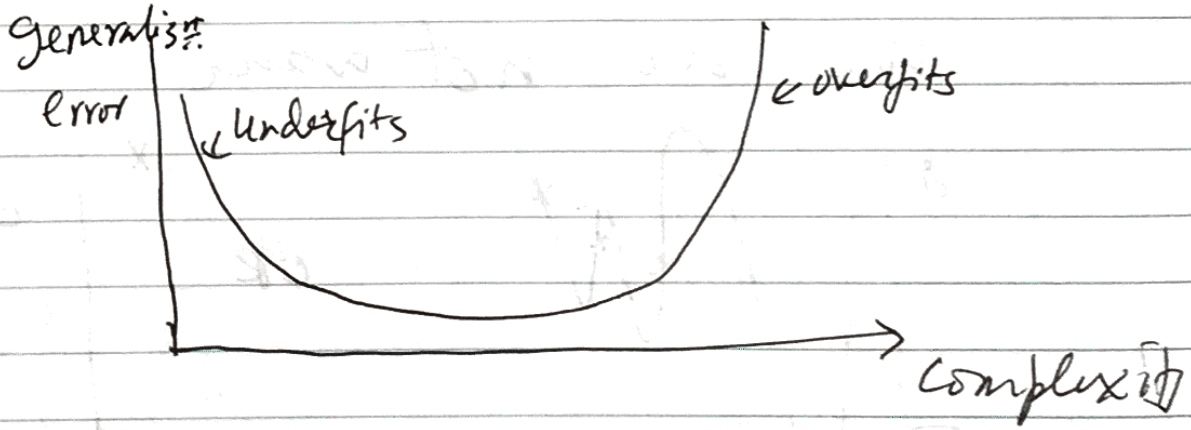
Reduces error on training set.

Generalization error

error on 'future' data

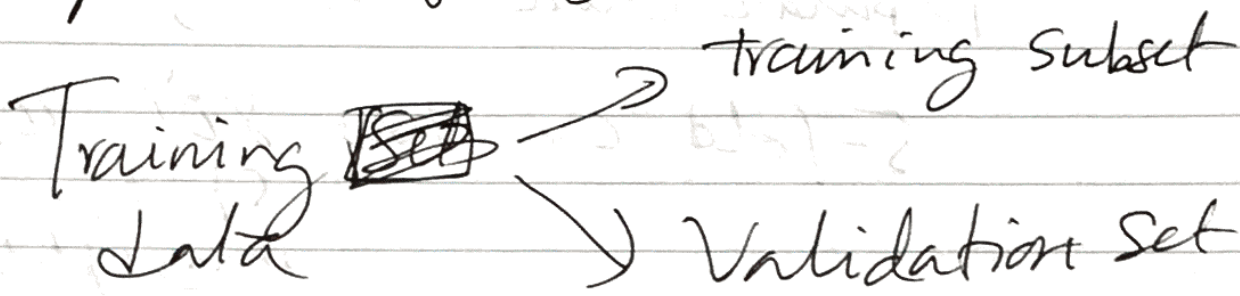
Hands on matlab exercise.

U shaped curve.



Common strategy to guard

against overfitting!

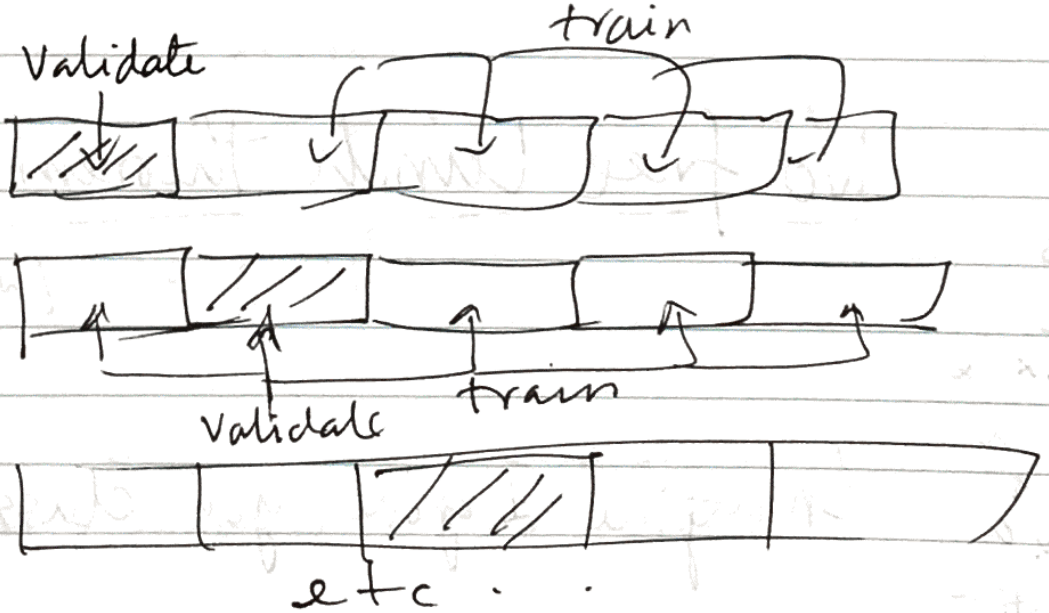


Often training set 80% validation/test set 20%.

If the dataset is small, often not enough statistics.

Popular solution: Cross validation (CV)

Split data k-fold



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

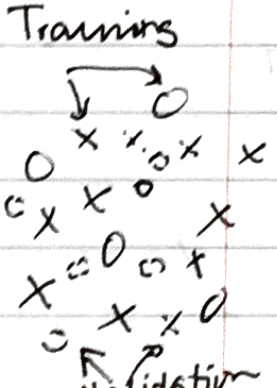
Choosing number of nearest neighbors in KNN.

" " " parameters in regression.

No free lunch theorem:

(Wolpert 1996)

Imagine I give you classification



task: $D = \{(x_i, y_i)\}_{i=1}^{N}$ is the training set

$y_i \in \{0,1\}$

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

For simplicity assume no x's are the same

Let $f : \mathcal{X} \to \mathcal{Y}$

On $\{x_1, \ldots, x_{N+M}\}$ there are

2^{N+M} choices for f. Consider

a prob. distr^{n} $P(f)$ that uniformly samples from these 2^{N+M} functions.

If D^f and D_V^f are generated by f, and algorithm training on D produces a function $\hat{f}$

Generalization error $\mathrm{err}(\hat{f}, D_V) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}(\hat{f}(x_{j+N}) \neq y_{j+N})$

$$E[\mathrm{err}(\hat{f}, D_V^f)] = \sum_f \mathrm{err}(\hat{f}, D_V^f)\, P(f) = ?$$

$$E[\mathrm{err}(\hat{f}, D_V^f)] = \frac{1}{2}$$

If algorithm $A \to f_{A,D^f}$ $\quad B \to f_{B,D^f}$

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 the structure of data to be processed.

Role of bias in learning!