Introduction to Machine Learning

Journées Calcul et Apprentissage

Aurélien Garivier
24-25 avril 2019
Table of contents

1. What is Machine Learning?
   Data and Learning Algorithms
   Classification Framework

2. Nearest-Neighbor Classification

3. Empirical Risk Minimization

4. Support Vector Machines

5. Neural Networks
What is Machine Learning?
Why Machine Learning?

Yann LeCun, Geoffrey Hinton et Yoshua Bengio reçoivent le prix Turing

Machine Learning for Science: State of the Art and Future Prospects

Abstract

Recent advances in machine learning have made it easier to analyze and predict outcomes in a wide range of fields, from medicine to finance. This has led to a new frontier of research in artificial intelligence.

Can we trust scientific discoveries made using machine learning?

Can we trust scientific discoveries made using machine learning?

Rice U. expert, key in creating AI systems that question their own predictions.

RICE UNIVERSITY
Where to learn more?
What do I need to practice ML? R
What do I need to practice ML? python
What do I need to practice ML?  

scikit-learn

Classification
- Identifying in which category an object should go.
- Applications: Spam detection, Image recognition.
- Algorithms: SVM, nearest neighbors, random forest...

Regression
- Predicting a continuous value associated with an object.
- Applications: Drug response, Stock prices.
- Algorithms: LR, ridge, regression, Lasso...

Clustering
- Automatic grouping of similar objects in a dataset.
- Applications: Customer segmentation, Grouping experimental outcomes.

Dimensionality reduction
- Reducing the number of random variables to consider.
- Applications: Visualization, Increased efficiency.
- Algorithms: PCA, feature selection, non-negative matrix factorization...

Model selection
- Selecting the best model for a given dataset.
- Applications: Cross-validation, Model selection.
- Goal: Improved accuracy via parameter tuning.
- Modules: GridSearch, Cross-validation, metrics...

Preprocessing
- Feature extraction and normalization.
- Applications: Data transformation such as text to use with machine learning algorithms.
- Modules: preprocessing, feature extraction.

News
- Ongoing development: What's new...

Community
- About us & contact...
- More Machine Learning related...

Who uses scikit-learn?
- AWeber
What is Machine Learning?

Data and Learning Algorithms

Classification Framework

Nearest-Neighbor Classification

Empirical Risk Minimization

Support Vector Machines

Neural Networks
What is Machine Learning?

• Algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions...

• ...rather than following strictly static program instructions: useful when designing and programming explicit algorithms is unfeasible or poorly efficient.

Within Artificial Intelligence

• evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

• AI: emulate cognitive capabilities of humans (big data: humans learn from abundant and diverse sources of data).

• a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".
Example: MNIST dataset
### Definition

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field of study that gives computers the ability to learn without being explicitly programmed</td>
<td>A computer program is said to learn from experience ( E ) with respect to some class of tasks ( T ) and performance measure ( P ) if its performance at tasks in ( T ), as measured by ( P ), improves with experience ( E ).</td>
</tr>
</tbody>
</table>
Machine Learning: Typical Problems

- spam filtering, text classification
- optical character recognition (OCR)
- search engines
- recommendation platforms
- speech recognition software
- computer vision
- bio-informatics, DNA analysis, medicine
- ...

For each of this task, it is possible but very inefficient to write an explicit program reaching the prescribed goal.

It proves much more successful to have a machine infer what the good decision rules are.
What is Statistical Learning?

= Machine Learning using statistics-inspired tools and guarantees

- Importance of probability- and statistics-based methods
  → Data Science (Michael Jordan)

- Computational Statistics: focuses in prediction-making through the use of computers together with statistical models (ex: Bayesian methods).

- Data Mining (unsupervised learning) focuses more on exploratory data analysis: discovery of (previously) unknown properties in the data. This is the analysis step of Knowledge Discovery in Databases.

- Machine Learning has more operational goals
  Ex: consistency → oracle inequalities
  Models (if any) are instrumental.
  ML more focused on correlation, less on causality (now changing).

- Strong ties to Mathematical Optimization, which furnishes methods, theory and application domains to the field
What is ML composed of?

Machine Learning:
- Supervised Learning: classification, regression
  - Decision Trees
  - SVM
  - Neural Networks
  - Ensemble Methods
  - Bagging
  - Boosting
  - Random Forest
- Unsupervised Learning
  - Clustering
  - Anomaly detection
  - Dimension reduction
  - Density estimation
  - Latent variables
- Reinforcement Learning
  - Bandits
  - MDP
  - Semi-supervised learning
- Rule Learning
  - Inductive logic programming
  - Association rule learning
- Representation learning
  - Bayesian networks
  - Latent variables
  - Density estimation
  - Dimension reduction

Model-based
- Sparse dictionary learning

Similarity / metric learning
- Recommender systems

11
Outline

What is Machine Learning?
- Data and Learning Algorithms
- Classification Framework

Nearest-Neighbor Classification

Empirical Risk Minimization

Support Vector Machines

Neural Networks
### What is a classifier?

A classifier is a function that maps data points from a feature space to a set of labels. Mathematically, it can be represented as:

\[
X \in \mathcal{M}_{n,p}(\mathbb{R}) \quad \text{and} \quad Y \in \mathcal{Y}^n
\]

where 
- \(X\) is an \(n\)-by-\(p\) matrix representing \(n\) examples of observations with \(p\) features.
- \(Y\) is a set of labels for the examples.

**Classifier** \(A_n\):

\[
h_n : X \rightarrow Y
\]

**Data:** 
- \(n\) examples = points of observations
- \(p\) features = characteristics measured for each example

\[
\begin{array}{c|c|c}
\text{Feature 1} & \text{Feature 2} & \text{Feature p} \\
\hline
x_1 & & y_1 \\
\hline
x_2 & & y_2 \\
\hline
\vdots & \ddots & \vdots \\
\hline
x_n & & y_n \\
\end{array}
\]
Data repositories

- Inside R: package datasets
- Inside python/scikitlearn: package sklearn.datasets
- UCI Machine Learning Repository

- Challenges: Kaggle, etc.
What is Machine Learning?

- Data and Learning Algorithms
- Classification Framework
- Nearest-Neighbor Classification
- Empirical Risk Minimization
- Support Vector Machines
- Neural Networks
Assumption

- The examples \((X_i, Y_i)_{1 \leq i \leq n}\) are iid samples of an unknown joint distribution \(\mathcal{D}\);
- The points to classify later are also independent draws of the same distribution \(\mathcal{D}\).

Hence, for every decision rule \(h : \mathcal{X} \to \mathcal{Y}\) we can define the risk

\[
L_{\mathcal{D}}(h) = \mathbb{P}_{(X, Y) \sim \mathcal{D}}(h(X) \neq Y) = \mathcal{D}\left( \{(x, y) : h(x) \neq y\} \right).
\]

The goal of the learning algorithm is to minimize the expected risk:

\[
R_n(\mathcal{A}_n) = \mathbb{E}_{\mathcal{D}^\otimes n}\left[ L_{\mathcal{D}}\left( \underbrace{\mathcal{A}_n((X_1, Y_1), \ldots, (X_n, Y_n))}_{\hat{h}_n} \right) \right]
\]

for every distribution \(\mathcal{D}\), using only the examples.
Signal and Noise

new york times bestseller
noise and the noise
the signal and the noise and the noise and the noise
why so many noise predictions fail—but some don’t the noise and the noise and the noise
nate silver the noise

www.guruprasad.net
Example: Character Recognition

<table>
<thead>
<tr>
<th>Domain set $\mathcal{X}$</th>
<th>Label set $\mathcal{Y}$</th>
<th>$28 \times 28$ images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint distribution $\mathcal{D}$</td>
<td>Label set $\mathcal{Y}$</td>
<td>${0, 1, \ldots, 9}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prediction function $h \in \mathcal{H} \subset \mathcal{Y}^\mathcal{X}$</th>
<th>Risk $R(h) = P_\mathcal{D}(h(X) \neq Y)$</th>
<th>?</th>
</tr>
</thead>
</table>

| Sample $S_n = \{ (X_i, Y_i) \}_{i=1}^n$ | MNIST dataset | |
| Empirical risk | | |

\[ L_S(h) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}\{h(X_i) \neq Y_i\} \]

<table>
<thead>
<tr>
<th>Learning algorithm</th>
<th>neural nets, boosting...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{A} = (\mathcal{A}_n)_n$, $\mathcal{A}_n : (\mathcal{X} \times \mathcal{Y})^n \rightarrow \mathcal{H}$</td>
<td></td>
</tr>
<tr>
<td>Expected risk $R_n(\mathcal{A}) = \mathbb{E}<em>n\left[ L</em>\mathcal{D}(\mathcal{A}_n(S_n)) \right]$</td>
<td></td>
</tr>
</tbody>
</table>
Two visions of $\mathcal{D}$

As a pair $(\mathcal{D}_x, k)$, where

- for $A \subset \mathcal{X}$, $\mathcal{D}_x(A) = \mathcal{D}(A \times \mathcal{Y})$ is the marginal distribution of $X$,
- and for $x \in \mathcal{X}$ and $B \subset \mathcal{Y}$,
  
  $$k(B|x) = \mathcal{D}(Y \in B|X = x)$$
  
  is (a version of) the conditional distribution of $Y$ given $X$.

As a pair $\left(\mathcal{D}_y, (\mathcal{D}(\cdot|y))_y\right)$, where

- for $y \in \mathcal{Y}$, $\mathcal{D}_y(y) = \mathcal{D}(\mathcal{X} \times y)$ is the marginal distribution of $Y$,
- and for $A \subset \mathcal{X}$ and $y \in \mathcal{Y}$,
  
  $$\mathcal{D}(A|y) = \mathcal{D}(X \in A|Y = y)$$
  
  is the conditional distribution of $X$ given $Y = y$. 
Two visions of $\mathcal{D}$

As a pair $(\mathcal{D}_x, k)$, where

- for $A \subset \mathcal{X}$, $\mathcal{D}_x(A) = \mathcal{D}(A \times \mathcal{Y})$ is the marginal distribution of $X$,
- and for $x \in \mathcal{X}$ and $B \subset \mathcal{Y}$, $k(B|x) = \mathcal{D}(Y \in B|X = x)$ is (a version of) the conditional distribution of $Y$ given $X$.

As a pair $(\mathcal{D}_Y, (\mathcal{D}(\cdot|y))_y)$, where

- for $y \in \mathcal{Y}$, $\mathcal{D}_Y(y) = \mathcal{D}(\mathcal{X} \times y)$ is the marginal distribution of $Y$,
- and for $A \subset \mathcal{X}$ and $y \in \mathcal{Y}$, $\mathcal{D}(A|y) = \mathcal{D}(X \in A|Y = y)$ is the conditional distribution of $X$ given $Y = y$. 
Consider binary classification \( \mathcal{Y} = \{0, 1\} \), \( \eta(x) := \mathcal{D}(Y = 1|X = x) \).

**Theorem**

The Bayes classifier is defined by

\[
  h^*(x) = \mathbb{1}\{\eta(x) \geq 1/2\} = \mathbb{1}\{\eta(x) \geq 1 - \eta(x)\} = \mathbb{1}\{2\eta(x) - 1 \geq 0\}.
\]

For every classifier \( h : \mathcal{X} \to \mathcal{Y} = \{0, 1\} \),

\[
  L_\mathcal{D}(h) \geq L_\mathcal{D}(h^*) = \mathbb{E}\left[ \min(\eta(X), 1 - \eta(X)) \right].
\]

The Bayes risk \( L^*_\mathcal{D} = L_\mathcal{D}(h^*) \) is called the **noise** of the problem.

More precisely,

\[
  L_\mathcal{D}(h) - L_\mathcal{D}(h^*) = \mathbb{E}\left[ |2\eta(X) - 1| \ 1\{h(X) \neq h^*(X)\} \right].
\]

Extends to \(|\mathcal{Y}| > 2\).
Nearest-Neighbor Classification
The Nearest-Neighbor Classifier

We assume that $\mathcal{X}$ is a metric space with distance $d$.

The nearest-neighbor classifier $\hat{h}_n^{\text{NN}} : \mathcal{X} \to \mathcal{Y}$ is defined as

$$\hat{h}_n^{\text{NN}}(x) = Y_l \text{ where } l \in \arg\min_{1 \leq i \leq n} d(x - X_i).$$

Typical distance: $L^2$ norm on $\mathbb{R}^d$: $\|x - x'\| = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2}$.

Buts many other possibilities: Hamming distance on $\{0, 1\}^d$, etc.
The most simple analysis of the most simple algorithm

A1. $\mathcal{Y} = \{0, 1\}$.
A2. $\mathcal{X} = [0, 1]^d$.
A3. $\eta$ is $c$-Lipschitz continuous:

$$\forall x, x' \in \mathcal{X}, |\eta(x) - \eta(x')| \leq c\|x - x'\|.$$}

**Theorem**

*Under the previous assumptions, for all distributions $\mathcal{D}$ and all $m \geq 1$

$$L_{\mathcal{D}}(\hat{h}_n^{NN}) \leq 2L_\mathcal{D}^* + \frac{3c\sqrt{d}}{n^{1/(d+1)}}.$$*
Proof Outline

• Conditioning: as \( I(x) = \arg \min_{1 \leq i \leq n} \| x - X_i \| \),

\[
L_D(\hat{h}_n^{NN}) = \mathbb{E} \left[ \mathbb{E} \left[ 1 \{ Y \neq Y_{I(X)} \} | X, X_1, \ldots, X_n \right] \right].
\]

• \( Y \sim B(p), \ Y' \sim B(q) \implies \mathbb{P}(Y \neq Y') \leq 2 \min(p, 1 - p) + |p - q|,

\[
\mathbb{E} \left[ 1 \{ Y \neq Y_{I(X)} \} | X, X_1, \ldots, X_n \right] \leq 2 \min \left( \eta(X), 1 - \eta(X) \right) + c \| X - X_{I(X)} \|.
\]

• Partition \( \mathcal{X} \) into \( |C| = T^d \) cells of diameter \( \sqrt{d} / T \):

\[
C = \left\{ \left[ \frac{j_1 - 1}{T}, \frac{j_1}{T} \right] \times \cdots \times \left[ \frac{j_d - 1}{T}, \frac{j_d}{T} \right], \quad 1 \leq j_1, \ldots, j_d \leq T \right\}.
\]

• 2 cases: either the cell of \( X \) is occupied by a sample point, or not:

\[
\| X - X_{I(X)} \| \leq \sum_{c \in C} 1 \{ X \in c \} \left( \frac{\sqrt{d}}{T} \mathbb{1} \bigcup_{i=1}^{n} \{ X_i \in c \} + \sqrt{d} \mathbb{1} \bigcap_{i=1}^{n} \{ X_i \notin c \} \right).
\]

• \( \implies \mathbb{E} [\| X - X_{I(X)} \|] \leq \frac{\sqrt{d}}{T} + \frac{\sqrt{d} T^d}{e n} \) and choose \( T = \left\lfloor n^{\frac{1}{d+1}} \right\rfloor \).
What does the analysis say?

- Is it loose? (sanity check: uniform $\mathcal{D}_X$)
- *Non-asymptotic* (finite sample bound)
- The second term $\frac{3c\sqrt{d}}{n^{1/(d+1)}}$ is *distribution independent*
- Does not give the trajectorial decrease of risk
- Exponential bound $d$ (cannot be avoided...)
  \[ \implies \text{curse of dimensionality} \]
- How to improve the classifier?
Let \( \mathcal{X} \) be a (pre-compact) metric space with distance \( d \).

**k-NN classifier**

\[
h^{k\text{NN}} : x \mapsto 1 \{ \hat{\eta}(x) \geq 1/2 \} = \text{plugin for Bayes classifier with estimator}
\]

\[
\hat{\eta}(x) = \frac{1}{k} \sum_{j=1}^{k} Y(j)(X)
\]

where

\[
d(X(1)(X), X) \leq d(X(2)(X), X) \leq \cdots \leq d(X(n)(X), X) .
\]
More neighbors are better?

$k = 1$
More neighbors are better?

$k = 3$
More neighbors are better? 

$k = 5$
More neighbors are better?

$k = 7$
More neighbors are better?

$k = 95$
Bias-Variance tradeoff

Risque de k-NN en fonction du nombre de voisins

$k$ vs. $\text{knn.est.risk}$
Risk bound

Let \( C_\epsilon \) be an \( \epsilon \)-covering of \( \mathcal{X} \):

\[
\forall x \in X, \exists x' \in C_\epsilon : d(x, x') \leq \epsilon .
\]

**Excess risk for k-nearest-neighbours**

If \( \eta \) is \( c \)-Lipschitz continuous: \( \forall x, x' \in \mathcal{X}, |\eta(x) - \eta(x')| \leq c \cdot d(x, x') \), then for all \( k \geq 2 \) and all \( n \geq 1 \):

\[
L(\hat{h}^{kNN}_n) - L(h^*) \leq \frac{1}{\sqrt{k \epsilon}} + \frac{2k|C_\epsilon|}{n} + 4c\epsilon
\]

\[
\leq \frac{1}{\sqrt{k \epsilon}} + (2 + 4c) \left( \frac{\alpha k}{n} \right)^{\frac{1}{d+1}} \quad \text{for } \epsilon = \left( \frac{\alpha k}{n} \right)^{\frac{1}{d+1}} ,
\]

\[
\leq (3 + 4c) \left( \frac{\alpha}{n} \right)^{\frac{1}{d+3}} \quad \text{for } k = \left( \frac{n}{\alpha} \right)^{\frac{2}{d+3}} .
\]
Room for improvement

- Lower bound? in $n^{-\frac{1}{d}}$.

- Margin conditions
  $\implies$ fast rates

- More regularity?
  $\implies$ weighted nearest neighbors

- Is regularity required everywhere?
  $\implies$ What matters are the balls of mass $\approx \frac{k}{n}$ near the decision boundary.

- 2 "parameters":
  - obvious: the number of neighbors $k$ (bias-variance tradeoff)
  - hidden: the distance $d$ (real problem)
Curse of dimensionality: No free lunch theorem

**Theorem**

Let $c > 1$ be a Lipschitz constant. Let $A$ be any learning algorithm for binary classification over a domain $\mathcal{X} = [0, 1]^d$. If the training set size is $n \leq (c + 1)^d / 2$, then there exists a distribution $D$ over $[0, 1]^d \times \{0, 1\}$ such that:

- $\eta(x)$ is $c$-Lipschitz;
- the Bayes error of the distribution is 0;
- with probability at least $1/7$ over the choice of $S_n \sim D^\otimes n$,

$$L_D(A(S_n)) \geq \frac{1}{8}.$$
Empirical Risk Minimization
Going empirical

Idea for every candidate rule $h$ in an hypothesis class $\mathcal{H}$, replace the unknown risk

$$L_D(h) = \mathbb{P}_{(X,Y) \sim D}(h(X) \neq Y)$$

by the computable empirical risk

$$L_{S_n}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{h(X_i) \neq Y_i\}$$

and use some uniform law of large numbers:

$$\mathbb{P}_D \left( \sup_{h \in \mathcal{H}} \left| L_{S_n}(h) - L_D(h) \right| > c \sqrt{\frac{D_{\mathcal{H}} \log(n) + \log \frac{1}{\delta}}{n}} \right) \leq \delta$$

where $D_{\mathcal{H}}$ is the Vapnik-Chervonenkis dimension of $\mathcal{H}$. 

32
Empirical Risk minimization

**Uniform law of large numbers:**

\[
P_D \left( \sup_{h \in \mathcal{H}} |L_{S_n}(h) - L_D(h)| > c \sqrt{\frac{D_{\mathcal{H}} \log(n) + \log \frac{1}{\delta}}{n}} \right) \leq \delta.
\]

→ **Empirical Risk Minimizer:**

\[
\hat{h}_n = \arg \min_{h \in \mathcal{H}} L_{S_n}(h).
\]

Good if

- the class \( \mathcal{H} \) is not too large
- the number \( n \) of examples is large enough

so as to ensure that 

\[
c \sqrt{\frac{D_{\mathcal{H}} \log(n) + \log \frac{1}{\delta}}{n}} \leq \epsilon.
\]

→ **Sample complexity** = number of examples required to have an \( \epsilon \)-optimal rule in the hypothesis class \( \mathcal{H} = O \left( \frac{D_{\mathcal{H}}}{\epsilon^2} \right) \).
**The class of halfspaces**

**Definition**

The class of linear (affine) functions on \( \mathcal{X} = \mathbb{R}^d \) is defined as

\[
L_d = \{ h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R} \}, \quad \text{where } h_{w,b}(x) = \langle w, x \rangle + b.
\]

The hypothesis class of halfspaces for binary classification is defined as

\[
\mathcal{H}S_d = \text{sign} \circ L_d = \left\{ x \mapsto \text{sign} \left( h_{w,b}(x) \right) : h_{w,b} \in L_d \right\}
\]

where \( \text{sign}(u) = \mathbb{1}\{u \geq 0\} - \mathbb{1}\{u < 0\} \). *Depth 1 neural networks.*

By taking \( \mathcal{X}' = \mathcal{X} \times \{1\} \) and \( d' = d + 1 \), we may omit the bias \( b \) and focus on functions \( h_w(x) = \langle w, x \rangle \).

**Property**

The VC-dimension of \( \mathcal{H}S_d \) is equal to \( d + 1 \).

Corollary: the class of halfspaces is learnable with sample complexity \( O\left( \frac{d+1 + \log(1/\delta)}{\epsilon^2} \right) \).
Realizable case: Learning halfspaces with a linear program solver

Realizable case: there exists $w^*$ such that $\forall i \in \{1, \ldots, n\}$, $y_i \langle w^*, x_i \rangle > 0$.

Then there exists $\bar{w} \in \mathbb{R}^d$ such that $\forall i \in \{1, \ldots, n\}$, $y_i \langle \bar{w}, x_i \rangle \geq 1$: if we can find one, we have an ERM.

Let $A \in M_{n,d}(\mathbb{R})$ be defined by $A_{i,j} = y_i x_{i,j}$, and let $\nu = (1, \ldots, 1) \in \mathbb{R}^m$. Then any solution of the linear program

$$\max_{w \in \mathbb{R}^d} \langle 0, w \rangle \quad \text{subject to} \quad Aw \geq \nu$$

is an ERM. It can thus be computed in polynomial time.
Rosenblatt’s Perceptron algorithm

**Algorithm:** Batch Perceptron

**Data:** training set $(x_1, y_1), \ldots, (x_n, y_n)$

1. $w_0 \leftarrow (0, \ldots, 0)$
2. $t \geq 0$
3. **while** $\exists i_t : y_{i_t} \langle w_t, x_{i_t} \rangle \leq 0$ **do**
   4. $w_{t+1} \leftarrow w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}$
   5. $t \leftarrow t + 1$
4. **return** $w_t$

Each update helps reaching the solution, since

$$y_{i_t} \langle w_{t+1}, x_{i_t} \rangle = y_{i_t} \left( w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}, x_{i_t} \right) = y_{i_t} \langle w_t, x_{i_t} \rangle + \|x_{i_t}\|.$$ 

Relates to a coordinate descent (stepsize does not matter).
Convergence of the Perceptron algorithm

**Theorem**

Assume that the dataset $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is linearly separable and let the *separation margin* $\gamma$ be defined as:

$$\gamma = \max_{w \in \mathbb{R}^d : \|w\| = 1} \min_{1 \leq i \leq n} \frac{y_i \langle w, x_i \rangle}{\|x_i\|}.$$

Then the perceptron algorithm stops after at most $1/\gamma^2$ iterations.

**Proof:** Let $w^*$ be such that $\forall 1 \leq i \leq n, \frac{y_i \langle w^*, x_i \rangle}{\|x_i\|} \geq \gamma$.

- If iteration $t$ is necessary, then
  $$\langle w^*, w_{t+1} - w_t \rangle = y_{it} \left\langle w^*, \frac{x_{it}}{\|x_{it}\|} \right\rangle \geq \gamma \quad \text{and hence} \quad \langle w^*, w_t \rangle \geq \gamma t.$$

- If iteration $t$ is necessary, then
  $$\|w_{t+1}\|^2 = \left\|w_t + y_{it} \frac{x_{it}}{\|x_{it}\|} \right\|^2 = \|w_t\|^2 + \frac{2y_{it} \langle w_t, x_{it} \rangle}{\|x_{it}\|} + y_{it}^2 \leq \|w_t\|^2 + 1 \leq 0$$
  and hence $\|w_t\|^2 \leq t$, or $\|w_t\| \leq \sqrt{t}$.

- As a consequence, the algorithm iterates at least $t$ times if
  $$\gamma t \leq \langle w^*, w_t \rangle \leq \|w_t\| \leq \sqrt{t} \quad \Rightarrow \quad t \leq \frac{1}{\gamma^2}.$$

In the worst case, the number of iterations can be exponentially large in the dimension $d$. Usually, it converges quite fast. If $\forall i, \|x_i\| = 1$, $\gamma = d(S, D)$ where $D = \{x : \langle w^*, x \rangle = 0\}$. 37
Computational difficulty of agnostic learning, and surrogates

**NP-hardness of computing the ERM for halfspaces**

Computing an ERM in the agnostic case is NP-hard.

See *On the difficulty of approximately maximizing agreements*, by Ben-David, Eiron and Long.

Since the 0-1 loss

\[ L_{S_n}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{y_i \langle w, x_i \rangle < 0\} \]

is intractable to minimize in the agnostic case, one may consider *surrogate* loss functions

\[ L_{S_n}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i \langle w, x_i \rangle) , \]

where the loss function \( \ell : \mathbb{R} \to \mathbb{R}^+ \)

- dominates the function \( \mathbb{I}\{u < 0\} \),
- and leads to a "simple" optimization problem (e.g. convex).
Logistic Regression

Regression model

- linear
- logistic
Logistic loss \( \mathcal{Y} = \{-1, 1\} \)

Statistics: "logistic regression":

\[
P_w(Y = y | X = x) = \frac{1}{1 + \exp(-y \langle w, x \rangle)}
\]

Convex minimization problem, can be solved by Newton’s algorithm (in small dimension) or stochastic gradient descent (in higher dimension).
Structural Risk minimization

What if $\mathcal{H} = \bigcup_{d=1}^{\infty} \mathcal{H}_d$, with $\mathcal{H}_d \subset \mathcal{H}_{d+1}$?

→ empirical risk minimization fails

→ structural risk minimization:

$$\hat{h}_n = \arg \min_{d \geq 1, h \in \mathcal{H}_d} L_{S_n}(h) + D_{\mathcal{H}_d} \log(n).$$
Support Vector Machines
Margin for linear separation

- Training sample $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$.
- Linearly separable if there exists a halfspace $h = (w, b)$ such that $\forall i, y_i = \text{sign}(\langle w, x_i \rangle + b)$.
- What is the best separating hyperplane for generalization?

**Distance to hyperplane**

If $\|w\| = 1$, then the distance from $x$ to the hyperplane $h = (w, b)$ is $d(x, \mathcal{H}) = |\langle w, x \rangle + b|$.

**Proof:** Check that $\min \{\|x - v\|^2 : v \in h\}$ is reached at $v = x - (\langle w, x \rangle + b)w$. 
Hard-SVM

Formulation 1:
\[
\arg \max_{(w,b):\|w\|=1} \min_{1 \leq i \leq m} \left| \langle w, x_i \rangle + b \right| \text{ such that } \forall i, y_i (\langle w, x_i \rangle + b) > 0 .
\]

Formulation 2:
\[
\min_{w,b} \|w\|^2 \text{ such that } \forall i, y_i (\langle w, x_i \rangle + b) \geq 1 .
\]

Remark: \( b \) is not penalized.

Proposition

The two formulations are equivalent.

Proof: if \((w_0, b_0)\) is the solution of Formulation 2, then \(\hat{w} = \frac{w_0}{\|w_0\|}, \hat{b} = \frac{b_0}{|w_0|}\) is a solution of Formulation 1: if \((w^*, b^*)\) is another solution, then letting \(\gamma^* = \min_{1 \leq i \leq m} y_i (\langle w, x_i \rangle + b)\) we see that \(\left( \frac{w^*}{\gamma^*}, \frac{b^*}{\gamma^*} \right)\) satisfies the constraint of Formulation 2, hence \(\|w_0\| \leq \|w^*\| = \frac{1}{\gamma^*}\) and thus \(\min_{1 \leq i \leq m} \left| \langle \hat{w}, x_i \rangle + \hat{b} \right| = \frac{1}{\|w_0\|} \geq \gamma^*\).
Sample Complexity

**Definition**

A distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$ is *separable with a $(\gamma, \rho)$-margin* if there exists $(w^*, b^*)$ such that $\|w^*\| = 1$ and with probability 1 on a pair $(X, Y) \sim \mathcal{D}$, it holds that $\|X\| \leq \rho$ and $Y(\langle w^*, X \rangle + b) \geq \gamma$.

Remark: by multiplying the $x_i$ by $\alpha$, the margin is multiplied by $\alpha$.

**Theorem**

For any distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$ that satisfies the $(\gamma, \rho)$-separability with margin assumption using a homogenous halfspace, with probability at least $1 - \delta$ over the training set of size $n$ the 0–1 loss of the output of Hard-SVM is at most

$$\sqrt{\frac{4(\rho/\gamma)^2}{n}} + \sqrt{\frac{2 \log(2/\delta)}{n}}.$$ 

Remark: depends on dimension $d$ only thru $\rho$ and $\gamma$. 

When the data is not linearly separable, allow \textit{slack variables} $\xi_i$: 

$$\min_{w, b, \xi} \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \quad \text{such that } \forall i, y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

$$= \min_{w, b} \lambda \|w\|^2 + L_{S_n}^{\text{hinge}} (w, b) \quad \text{where } \ell_{\text{hinge}}(u) = \max(0, 1 - u).$$

\textbf{Theorem} 

Let $D$ be a distribution over $B(0, \rho) \times \{\pm 1\}$. If $A_n(S_n)$ is the output of the soft-SVM algorithm on the sample $S$ of $D$ of size $n$, 

$$\mathbb{E} \left[ L_{D}^{0-1} (A_n(S_n)) \right] \leq \mathbb{E} \left[ L_{D}^{\text{hinge}} (A_n(S_n)) \right] \leq \inf_{u} L_{D}^{\text{hinge}} (u) + \lambda \|u\|^2 + \frac{2\rho^2}{\lambda n}. $$

For every $B > 0$, setting $\lambda = \sqrt{\frac{2\rho^2}{B^2n}}$ yields:

$$\mathbb{E} \left[ L_{D}^{0-1} (A_n(S_n)) \right] \leq \mathbb{E} \left[ L_{D}^{\text{hinge}} (A_n(S_n)) \right] \leq \inf_{w: \|w\| \leq B} L_{D}^{\text{hinge}} (w) + \sqrt{\frac{8\rho^2B^2}{n}}.$$
Margin maximization leads to

\[
L_{S_n}^{\text{hinge}}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \max \left\{ 0, 1 - y_i \langle w, x_i \rangle \right\},
\]

a convex but non-smooth minimization problem, used with a penalization term \( \lambda \|w\|^2 \).
Dual Form of the SVM Optimization Problem

To simplify, we consider only the homogeneous case of hard-SVM. Let

$$g(w) = \max_{\alpha \in [0, +\infty)^n} \sum_{i=1}^{n} \alpha_i (1 - y_i \langle w, x_i \rangle) = \begin{cases} 0 & \text{if } \forall i, y_i \langle w, x_i \rangle \geq 1, \\ +\infty & \text{otherwise} \end{cases}.$$ 

Then the hard-SVM problem is equivalent to

$$\min_{w: \forall i, y_i \langle w, x_i \rangle \geq 1} \frac{1}{2} \|w\|^2 = \min_{w} \frac{1}{2} \|w\|^2 + g(w)$$

$$= \min_{w} \max_{\alpha \in [0, +\infty)^m} \frac{1}{2} \|w\|^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i \langle w, x_i \rangle)$$

$$\overset{\text{min-max thm}}{=} \max_{\alpha \in [0, +\infty)^n} \min_{w} \frac{1}{2} \|w\|^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i \langle w, x_i \rangle).$$

The inner min is reached at $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and can thus be written as

$$\max_{\alpha \in \mathbb{R}^n, \alpha \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{1 \leq i, j \leq n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle.$$
Still for the homogeneous case of hard-SVM:

**Property**

Let $w_0$ be a solution of and let $I = \{ i : |\langle w_0, x_i \rangle| = 1 \}$. There exist $\alpha_1, \ldots, \alpha_n$ such that

$$w_0 = \sum_{i \in I} \alpha_i x_i .$$

The dual problem involves the $x_i$ only thru scalar products $\langle x_i, x_j \rangle$.

It is of size $n$ (independent of the dimension $d$).

These computations can be extended to the non-homogeneous soft-SVM → **Kernel trick**.
Numerically solving Soft-SVM

\[ f(w) = \frac{\lambda}{2} \|w\|^2 + L_{S}^{\text{hinge}}(w) \] is \( \lambda \)-strongly convex.

\[ \rightarrow \] Stochastic Gradient Descent with learning rate \( 1/(\lambda t) \). Stochastic subgradient of \( L_{S}^{\text{hinge}}(w) \) : 
\[ v_t = -y_{l_t}x_{l_t} \mathbb{I}\{y_{l_t}\langle w, x_{l_t} \rangle < 1\} . \]

\[ w_{t+1} = w_t - \frac{1}{\lambda t} (\lambda w_t + v_t) = \frac{t - 1}{t} w_t - \frac{1}{\lambda t} v_t = -\frac{1}{\lambda t} \sum_{i=1}^{t} v_t . \]

**Algorithm: SGD for Soft-SVM**

1. Set \( \theta_0 = 0 \)
2. for \( t = 0 \ldots T - 1 \) do
3. \[ \text{Let } w_t = \frac{1}{\lambda t} \theta_t \]
4. \[ \text{Pick } l_t \sim \mathcal{U}(\{1, \ldots, n\}) \]
5. if \( y_{l_t}\langle w_t, x_{l_t} \rangle < 1 \) then
6. \[ \theta_{t+1} \leftarrow \theta_t + y_{l_t}x_{l_t} \]
7. else
8. \[ \theta_{t+1} \leftarrow \theta_t \]
9. return \( \bar{w}_T = \frac{1}{T} \sum_{t=0}^{T-1} w_t \)
Neural Networks
One-layer network

Src: http://insanedev.co.uk/open-cranium/
One-layer network

Src: [Tufféry, Data Mining et Informatique Dcisionnelle]
One-layer network
Two-layer network

Src: [Tuffery, Data Mining et Informatique Decisionnelle]
Profound ideas and tricks

- Convolutional networks
- Max-pooling
- Dropout
- Data augmentation
- GANs
- Representation learning
- Self-learning (ex: classify against rotations)
The three main theoretical challenges of deep learning

- **Expressive power of DNN**: why are the function we are interested in so well approximated by (deep convolutive) neural networks?

- **Success of naive optimisation**: why does gradient descent lead to a good local minimum?

- **Generalization miracle**: why is there no overfitting with so many parameters?