

Chapter 1 - Deep Learning

Evaluation

- Exams at the end of each speaker's classes: 50% (alone) 每个老师课程结束都有考试，以多项选择题形式考察，总共有五次考试，考试权重按课程数量来分布
- Project: 50% (groups of 2 or 3)
Grades will take into account the quality of your presentation and report, how you tackled the task, and a bonus for performance.

- Deep Learning Written Exam 20%
- Reinforcement Learning Written Exam 20%
 - (Multiple Choice Questions and open questions)
- Paper presentation 10%
- Project 50%
 - Kaggle project (or else) working in collaboration in pairs to solve a task
 - Choice of a project among a pool of propositions (object detection, text analysis, ...)
 - Submit a report for last session of 2-4 pages
 - 15 minutes presentation at last session (same as written exam)
 - Grades take into account: technical difficulty, understanding of the problem, understanding of the solution, quality of the report, quality of the presentation

Machine Learning Basics

Linear Algebra

Notations

- **Scalars:** a single number, example n the number of hidden neurons.
- **Vectors:** an array of number, example of a \mathbb{R}^n vector elements $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
- **Matrices:** a 2 dimensional array of numbers, example of a matrix
$$\mathbf{A} \in \mathbb{R}^{m \times n} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}$$
- **Tensors:** a n dimensional array of numbers, example $\mathbf{A} \in \mathbb{R}^{m \times k \times p}$ a 3 dimensional tensor

Vectors and matrices operations

- Matrix transposition: \mathbf{A}^T transposed of \mathbf{A} defined as $(\mathbf{A}^T)_{i,j} = A_{j,i}$
- Matrix multiplication: Let $\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{B} \in \mathbb{R}^{k \times n}$, $\mathbf{C} = \mathbf{AB}$ is defined for each \mathbf{C} element as $C_{i,j} = \sum_k A_{i,k} B_{k,j}$ with $\mathbf{C} \in \mathbb{R}^{m \times n}$
- Transposed matrix multiplication: $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- Matrix-vector multiplication: $\mathbf{Ax} = \sum_i x_i \mathbf{A}_{:,i}$
- Identity matrix: A square matrix that preserve any vectors it is multiplied with. For vectors of size n , the identity matrix \mathbf{I}_n is defined by $\mathbf{I}_n \mathbf{x} = \mathbf{x}$
- Matrix inverse: \mathbf{A}^{-1} inverse of \mathbf{A} defined by $\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}_n$

Norms

- A **norm** is a function f that measures the size of vectors with the following properties:

- **Positive definiteness:** $f(x) = 0 \implies \mathbf{x} = 0$
- **Triangle inequality:** $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$
- **Absolute homogeneity** $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$

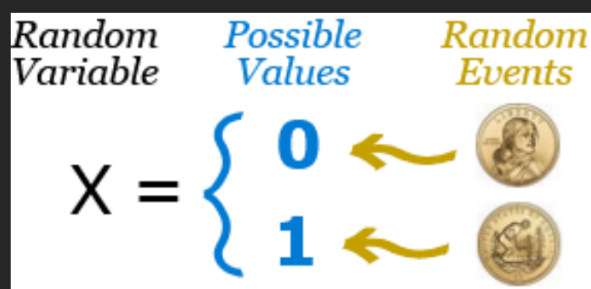
- The L^p norm is defined for a vector \mathbf{x} as

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- **Euclidian norm** is the L^2 norm, noted $\|\mathbf{x}\|$ and equal to $\sqrt{\mathbf{x}^T \mathbf{x}} = \sqrt{\sum_i x_i^2}$. Generally, the squared Euclidian norm is used
- **Manhattan norm** is the L^1 norm and is used when the difference between zero and nonzero elements is important
- **Max norm** is defined as $\|\mathbf{x}\|_\infty = \max_i |x_i|$

Random Variables

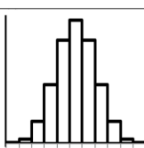
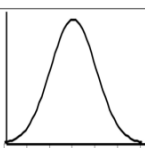
- A **random variable** is a variable that can take different values randomly
- Notation: a random variable \mathbf{x} can take different x values
- Random variables can be **discrete**, like the number of a dice, or **continuous**, like the humidity in the air.



Probability

Probability Distribution

- A p probability distribution is a **Probability Mass Function (PMF)** for discrete random variables and a **Probability Density Function (PDF)** for continuous random variables
- p has to satisfy the following properties:
 - p domain describe all possible states of \mathbf{x}
 - $\forall x \in \mathbf{x}, p(x) \geq 0$
 - $\int p(x) dx = 1$

Discrete	Continuous
	
Probability Mass Function	Probability Density Function
Count, Sum, Proportion	Integration
$P(X = x) = f(x)$	$P(X=x) = \int f(x). dx$
CMF, PMF = Sum, Difference	CDF, PDF = Integrate, Differentiate

Marginal and Conditional Probabilities

We have two random variables \mathbf{x} and \mathbf{y} and we know the probability distribution $p(x, y)$

- Marginal probability:**

$$\forall x \in \mathbf{x}, p(\mathbf{x} = x) = \int p(x, y) dy$$

- Conditional probability:**

$$p(\mathbf{y} = y | x) = \frac{p(\mathbf{y} = y, \mathbf{x} = x)}{p(\mathbf{x} = x)}$$

- Chain rule of Conditional Probabilities:**

$$p(x^{(1)}, \dots, x^{(n)}) = p(x^{(1)}) \prod_{i=2}^n p(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$

Independance and Conditional Independance

- \mathbf{x} and \mathbf{y} are **independent**, $\mathbf{x} \perp \mathbf{y}$, if

$$\forall x \in \mathbf{x}, y \in \mathbf{y}, p(\mathbf{x} = x, \mathbf{y} = y) = p(\mathbf{x} = x)p(\mathbf{y} = y)$$

- \mathbf{x} and \mathbf{y} are **conditionally independent** We have two random variables \mathbf{x} and \mathbf{y} and we know the probability distribution $p(x, y)$

- Marginal probability:**

$$\forall x \in \mathbf{x},$$

$$\forall x \in \mathbf{x}, p(\mathbf{x} = x) = \int p(x, y) dy$$

$$p(\mathbf{x} = x, \mathbf{y} = y | \mathbf{z} = z) =$$

- Conditional probability:**

$$p(\mathbf{y} = y | x) = \frac{p(\mathbf{y} = y, \mathbf{x} = x)}{p(\mathbf{x} = x)}$$

- Chain rule of Conditional Probabilities:**

$$p(x^{(1)}, \dots, x^{(n)}) = p(x^{(1)}) \prod_{i=2}^n p(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$

Expectation, Variance and Covariance

- **Expectation** of a function $f(x)$ given $p(x)$ is the average value of f on x

$$\mathbb{E}_{x \sim p}[f(x)] = \int p(x) f(x) dx$$

- **Variance** of $f(x)$ measure how the values of f varies from its average

$$\text{Var}(f(x)) = \mathbb{E} [(f(x) - \mathbb{E}[f(x)])^2]$$

and the **standard deviation** is the square root of the variance

- **Covariance** of two random variables provides information of how much two values are linearly related

$$\text{Cov}(f(x), g(y)) = \mathbb{E} [(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])]$$

Machine Learning

Objective

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 .

Task T : classification, regression, translation, generation, anomaly detection, ...

Performance measure P : specific to the tasks such as accuracy for classification. It is measured on a **test set**.

Experience E : two main categories supervised and unsupervised
Supervised learning: a dataset of points associated with a label or a target
Unsupervised learning: a dataset of points without labels or targets

Mathematically

- A dataset of m points and k features can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{m \times k}$.
- In case of supervised learning \mathbf{X} is associated with a vector of labels \mathbf{y} and we aim to learn a joint distribution $p(\mathbf{x}, y)$ to infer

$$p(x|\mathbf{y}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')}$$

- The goal of machine learning is to find a function \tilde{f} such as \tilde{f} associate each \mathbf{x} to the best approximation of \mathbf{y} and that it is capable to generalize to unseen data.
- \tilde{f} can be parameterized by a set of parameters θ

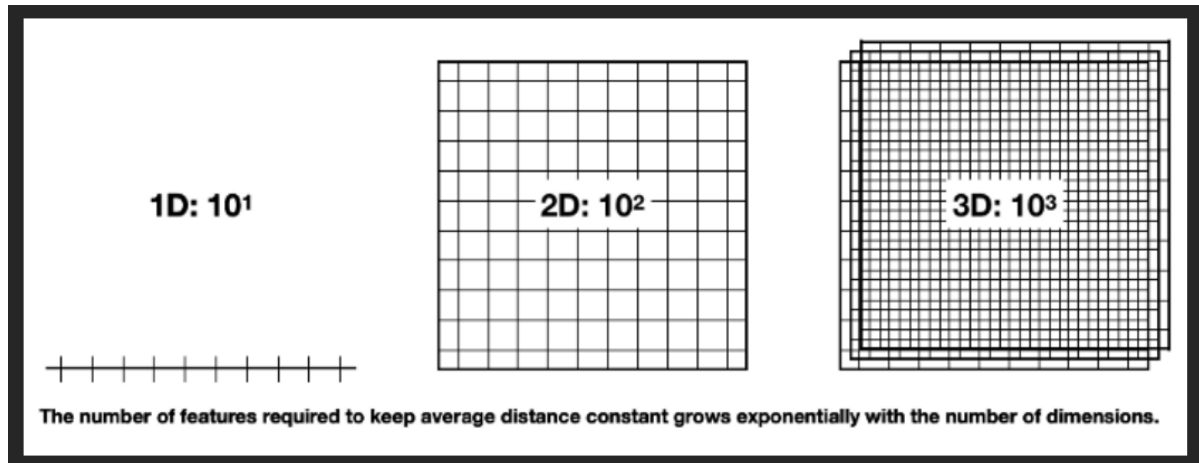
The capacity of a model

The main challenge of a machine learning model is **generalization** to unseen data estimated on *test* data after training on *training* data

Overfitting occurs when the gap between training error and test error is too large and **underfitting** when the training error is too low

The **capacity** of a model is the range of functions it is able to learn and control how likely the model can overfit or underfit

Curse of Dimensionality



Estimators

- The best parameter θ is unknown and we seek to estimate it through an **estimator** $\hat{\theta}$
- A **point estimator** is any function of the data \mathbf{X}

$$\hat{\theta} = g(\mathbf{X})$$

- The **bias** of an estimator is

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

An estimator is unbiased if $\text{bias}(\hat{\theta}) = 0$

- The **variance** of an estimator is

$$\text{Var}(\hat{\theta})$$

Maximum Likelihood Estimation

- To make a good estimator one of the most common principle is **maximum likelihood**.
- Consider $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ and $p(\mathbf{x}; \theta)$ a parametric family of probability distributions that maps for each x a real number estimation the probability $p_{data}(x)$

- The **maximum likelihood estimator** is

$$\begin{aligned}\theta_{\text{ML}} &= \arg \max_{\theta} p_{\text{model}}(\mathbf{X}; \theta) \\ &= \arg \max_{\theta} \prod_{i=1}^m p_{\text{model}}(\mathbf{x}^{(i)}; \theta)\end{aligned}$$

- The maximum log-likelihood estimator is often used to remove the product:

$$\theta_{\text{ML}} = \arg \max_{\theta} \sum_{i=1}^m \log(p_{\text{model}}(\mathbf{x}^{(i)}; \theta))$$

Unsupervised Machine learning: Data Visualisation and Clustering

Common Characteristics for Data

Redundancy: Original dimensions of the data are often higher than what is needed

Clusterable data: Groups can be defined to describe the input data

Structured data: Data can follow a given structure (Shape, Manifold etc...)

Projection through PCA(Principal Component Analysis)

Definition

Supposing we have a dataset $X \in \mathbb{R}^{n \times m}$, PCA projection consists in finding a new orthonormal coordinate system, such as the first coordinates retain most of the variance. I.e., with Y the projected data, the first j component $Y_{1...j}$ maximizes the variance.

Definition

- The orthogonal projection of a vector x on a normalized vector u is $P_u(x) = (x^T u)u$
- We thus look for directions u that maximize the variance of the projection $Var(||P_u(x_i)||) = \frac{1}{n-1} \sum_i (x_i^T u - \bar{x}^T u)^2 = u^T \Sigma u$
- I.e. we want to find $\arg \max_u u^T \Sigma u$, s.t. $||u|| = 1$

PCA as a constraint problem

$$f(x, y) = x^2 + y^2, g(x, y) = xy - 1$$

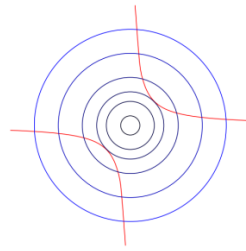


Figure: Optimisation sous contraintes avec le lagrangien, *Agathe Herrou, CNRS*

- **Objective:** While staying on the line $g = 0$, we want to maximize f . We thus look for stationary point (local maxima) of f when walking on g .
- It happens on the contour line of f (i.e., $f(x, y) = c$) which occurs formally when the lines defined by $g = 0$ and $f = c$ are parallel.

- f and g are parallel when ∇f and ∇g are parallel.
- i.e. when $\exists \lambda$, s.t. $\nabla f = \lambda \nabla g$
- Lagrangian function : $L(x, \lambda) = f(x) + \lambda g(x)$
- Under the condition of $-\nabla_x^2 L$ being positive semi-definite, finding x s.t. $\nabla_\lambda L = \nabla_x L = 0$ thus fulfills the constraint objective.

- In our case, the Lagrangian is :

$$L(u, \lambda) = u^T \Sigma u + \lambda(1 - u^T u)$$

•

$$\nabla_u L = 0 \Leftrightarrow \Sigma u = \lambda u$$

$$\nabla_\lambda L = 0 \Leftrightarrow u^T u = 1$$

- $\Sigma u = \lambda u \Rightarrow u^T \Sigma u = \lambda$, so to maximize $u^T \Sigma u$ we need to find the eigenvector associated to the highest eigenvalue

We can repeat this process, adding the orthogonality constraint d time

Or, using the spectral theorem, we know that σ , being a positive-semidefinite matrix, has d orthonormal eigenvectors, with d the number of nonnull eigenvalues.

PCA computation: summary

Center data (Why?)

Use singular value decomposition on $X^{**T}X$, which will give you eigenvalues and associated eigenvectors.

Order the eigen vector w.r. the eigenvalues (from highest to lowest), and project the centered data on the d first eigenvectors

PCA limitations

Performs linear transformations on the data (linear projection)

Mean of the data and covariance matrix are supposed to be enough to reduce dimensionality

Assume that large variance is an interest criterion

Kernel PCA

使用Kernel把数据升维，来发现更多的高维特征，之后把数据恢复维度

- Idea: data are separable in higher dimension
- **Theoretically** relies on a non-linear transformation

$$\phi : \mathbf{R}^m \rightarrow \mathbf{k}, \quad x \rightarrow \phi(x)$$

with $k > m$. This transformation (also called mapping) is complicated to find, instead it is easier to define a kernel function $k(x, y) = \phi(x)^T \phi(y)$

Kernel PCA: quick theoretical summary

- We apply the same logic as with PCA but on the $\phi(x_i)$ (feature space) instead of just x_i
- we obtain $\Sigma = \frac{1}{n-1} \sum_i^n \phi(x_i) \phi(x_i)^T$
- Solving $\Sigma u = \lambda u$ we obtain

$$u = \sum_i \frac{1}{\lambda(n-1)} (\phi(x_i)^T u) \phi(x_i) = \sum_i \alpha_i \phi(x_i)$$

- Sparing you the details we obtain at the end :

$$\forall j \in 1 \dots n : \frac{1}{n-1} \sum_{s=1}^n \alpha_s \sum_i k(x_j, x_i) k(x_i, x_s) = \lambda \sum_i \alpha_i k(x_i, x_j)$$

特征空间映射：通过 $\phi(x)$ 将数据从原始空间映射到高维空间。

协方差矩阵：在特征空间中构建协方差矩阵 Σ ，它表示高维空间中数据点的相关性。

特征值问题：通过求解 $\Sigma u = \lambda u$ ，找到主成分方向 u 。

核方法简化计算：通过核函数 $k(x_i, x_j)$ 来避免显式构造映射 $\phi(x)$ ，直接在低维空间中进行高维空间的计算。

Kernel PCA: quick theoretical summary

- Noting $\mathbb{K} = \{k(x_i, x_j)\}_{i,j \in [1,n]^2}$ and $\alpha = \{\alpha_i\}_{i \in [1,n]}$, we have :

$$\mathbf{K}\alpha = \lambda(n-1)\alpha$$

- The constraint $u^T u = 1$ gives, by replacing u with its expression

$$\alpha^T \alpha = \frac{1}{\lambda(n-1)} = \frac{1}{c}$$

- Solving these equations is thus equivalent to find the eigen vectors of \mathbf{K} and divide them by $c_i = \lambda_i(n-1)$
- We can then find for each component j of a sample x $y_j = \sum_i \alpha_i^j k(x, x_i)$

核矩阵 \mathbf{K} : 通过核函数计算的数据点之间的相似性矩阵。

特征值问题: 通过求解 $\mathbf{K}\alpha = \lambda(n-1)\alpha$, 找到核矩阵的特征值和特征向量。

归一化: 确保特征向量满足单位长度的约束, 通过除以特征值 $c_i = \lambda_i(n-1)$ 完成。

投影计算: 通过核函数和特征向量计算样本在特征空间中的分量。

Kernel PCA: which kernel function to use?

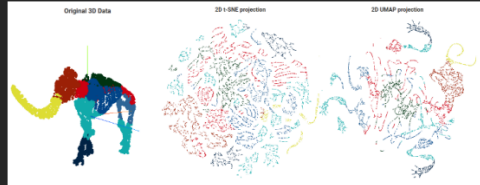
- k must satisfy the **Mercier's condition**
- In finite dimension, it is simplified by having \mathbf{K} semi-definite positive
- Commonly used kernels are the gaussian kernel ($k(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}}$) or the polynomial one $k(x, y) = (x^T y)^a$

Mercier's condition is a criterion used in plasma physics to ensure the stability of a plasma in a magnetic confinement system by evaluating the balance between pressure and magnetic field curvature.

在这里, Mercier条件提到了核函数 k 必须满足的一个条件, 即核矩阵 \mathbf{K} 必须是**半正定** (semi-definite positive)。这意味着, 对于任意输入数据, 核矩阵 \mathbf{K} 的所有特征值都必须是非负的。这个条件保证了核PCA (或其他核方法) 能够正确运行。

Mercier条件原本出现在等离子体物理中, 但在这里被类比用于机器学习领域, 表明核矩阵必须满足某些数学上的稳定性条件。

Other Projections approaches: t-SNE and UMAP



- Figure: <https://pair-code.github.io/understanding-umap/>
- t-SNE apply for each data point a Gaussian kernel on this point and its neighbors and normalize the values to obtain probabilities values. Modeling the probabilities of the lower-dimensioned space with a student kernel, t-SNE minimizes the KL divergence between the probabilities in the lower dimension and probabilities in the higher one.
 - UMAP relies on topological analysis of the high-dimension data to define a structure of the data. Optimize the projected space to reproduce the structure of the original space.

Clustering Algorithms : K-means

Algorithm 1: K-means Clustering

Input: Data points $X = \{x_1, x_2, \dots, x_n\}$, number of clusters k

Output: Cluster centroids $C = \{c_1, c_2, \dots, c_k\}$, cluster assignments for each point

Initialize C with k random points from X ;

repeat

foreach $x_i \in X$ **do**

Assign x_i to the nearest centroid c_j ;

foreach $c_j \in C$ **do**

Update c_j to be the mean of all points assigned to it;

until convergence;

return C , cluster assignments;

Deep Neural Networks

Perceptron

- A perceptron is an algorithm for supervised learning of binary classifiers (two classes).
- Suppose we have a dataset $\mathbf{X} \in \mathbb{R}^{n \times m}$ associated with a vector of labels $\mathbf{y} \in \{0, 1\}^n$
- It learns a function \tilde{f} parametrized by a vector of weights $\mathbf{w} \in \mathbb{R}^m$ and a bias b such as:

$$\tilde{f}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

Gradient Descent for Perceptron

Algorithm 2: Gradient Descent for Perceptron

Input: Data points $X = \{x_1, x_2, \dots, x_n\}$, labels $Y = \{y_1, y_2, \dots, y_n\}$, learning rate η , number of epochs T

Output: Weights w , bias b

Initialize w and b randomly;

for $t = 1$ **to** T **do**

foreach $(x_i, y_i) \in (X, Y)$ **do**

Compute $\hat{y}_i = wx_i + b$;

Compute loss $L = \frac{1}{2}(\hat{y}_i - y_i)^2$;

Compute gradients $\frac{\partial L}{\partial w} = (\hat{y}_i - y_i)x_i$;

Compute gradients $\frac{\partial L}{\partial b} = (\hat{y}_i - y_i)$;

Update $w := w - \eta \frac{\partial L}{\partial w}$;

Update $b := b - \eta \frac{\partial L}{\partial b}$;

Perceptron limitation

A perceptron cannot separate non linear data

Perceptron limitations mitigation : Adaboost

Let's $\{h\}$ be a set of perceptions (weeak classifiers), $H = \text{sign}(\sum h(x))$, we want to find the right h to get the best H possible.

这个问题讨论的是通过 **Adaboost** 来缓解感知

机 (Perceptron) 模型的局限性。

解释：

- $\{h\}$ 表示一组弱分类器，感知机是其中一个例子。这些弱分类器单独效果较差，但可以组合起来。
- $H = \text{sign}(\sum h(x))$ 是通过对多个弱分类器的输出进行加权组合，形成最终的强分类器 H 。通过计算各个弱分类器的结果 $h(x)$ ，然后求其符号得到最终分类结果。
- **Adaboost** 的目标是通过迭代加权，找到合适的弱分类器 h ，使得组合后的强分类器 H 达到最优的分类性能。

Adaboost 通过加大对错误分类样本的权重，逐步提高整体分类性能，从而缓解感知机单一模型的局限性。

Adaboost Algorithm

Input: \mathbf{H} : a chosen class of "weak" binary classifiers

Output: $F_t = \text{sign}(H_t)$

Initialize: $w_1(i) = \frac{1}{n}$, $H_0 = 0$;

for $t = 1$ **to** T **do**

$h_t = \arg \min_{h \in \mathbf{H}} \epsilon_t(h)$;

where $\epsilon_t(h) = \sum_{i \sim w_t} [h(x_i) \neq y_i]$;

 Choose α_t ;

 Update w_{t+1} ;

$H_t = H_{t-1} + \alpha_t h_t$;

end

Output: $F_T = \text{sign}(H_T)$;

Adaboost 是一种迭代算法，通过组合多个弱分类器来形成一个强分类器。每一轮迭代中，Adaboost 选择一个错误率最小的弱分类器，并给它分配一个权重 α_t 。同时，根据当前分类器的表现，调整样本的权重分布，使得下一轮更加关注当前分类错误的样本。

这样，强分类器 F_T 是所有弱分类器的加权组合，可以显著提高分类性能。

Updating parameters

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$w_{t+1} = \frac{w_t(I)e^{-\alpha_t y_i h_t(x_i)}}{Z_{t+1}}, \text{ } Z \text{ is chosen so that the sum of } w \text{ is equal to } 1$$

α_t 决定了每个弱分类器在最终强分类器中的权重，错误率低的分类器权重大，错误率高的分类器权重小。

w_{t+1} 更新样本权重，错误分类的样本权重增加，以便让下一轮更加关注这些分类错误的样本。

Z_{t+1} 是一个归一化因子，确保权重之和为 1。