Neural Networks

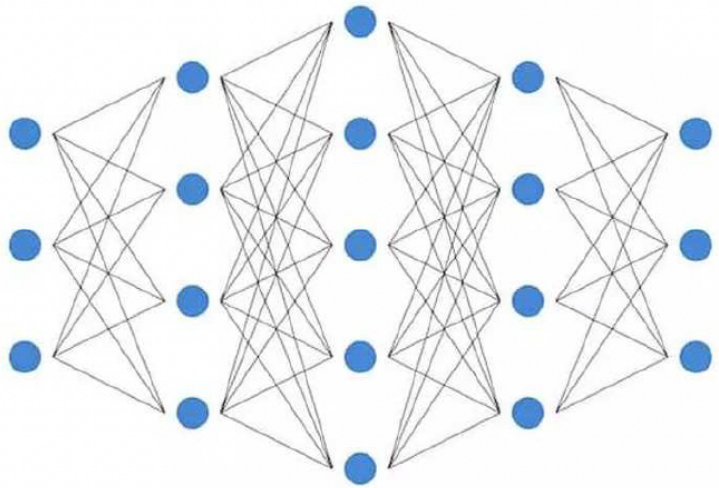
Jana de Wiljes

13. Juni 2022

Motivation

Data

Different parametrisations possible



How can we efficiently process the data, to learn functions with a high prediction ability?

Problem setting

Goal: Approximate function f, that describes the link between two random variables X and Y which have the joint distribution $\pi(z) = \pi(x, y)$

Choice of parametrisation:

- choose model class $\mathcal{H}$
- and appropriate loss functional $l(y, h(x))$

Expected Risk

For $h \in \mathcal{H}$ we define the expected Risik as follows

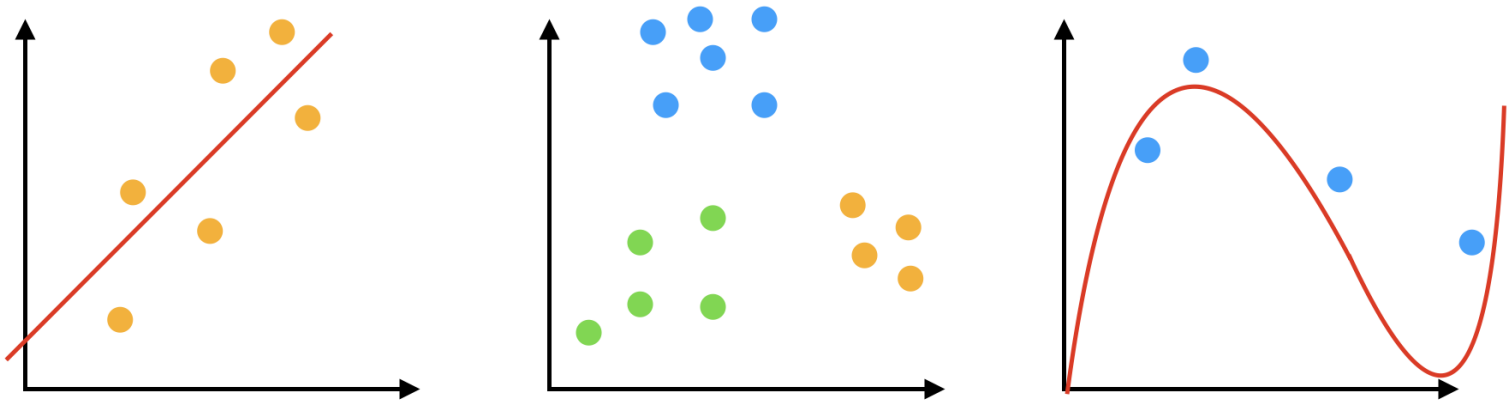
$$R(h) = \int_{\mathbf{Z}} l(y, h(x))\pi(z)dz \tag{1}$$

Approach: Want to find $h \in \mathcal{H}$ so that

$$h^* = \arg\min_{h \in \mathcal{H}} R(h) \tag{2}$$

Empirical Risk

Given in practice: independent and identical distributed Samples
$S = \{(x_i, y_i)\}_{i=1}^{N}$ with $(x_i, y_i) \sim \pi(x, y)$ for $i \in \{1, \ldots, N\}$



Empirical Risk

For a given sample set S we define the corresponding empirical risk as follows:

$$R_S(h) = \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i))$$

Empirical Risk-Minimizer

Empirical Risk-Minimizer

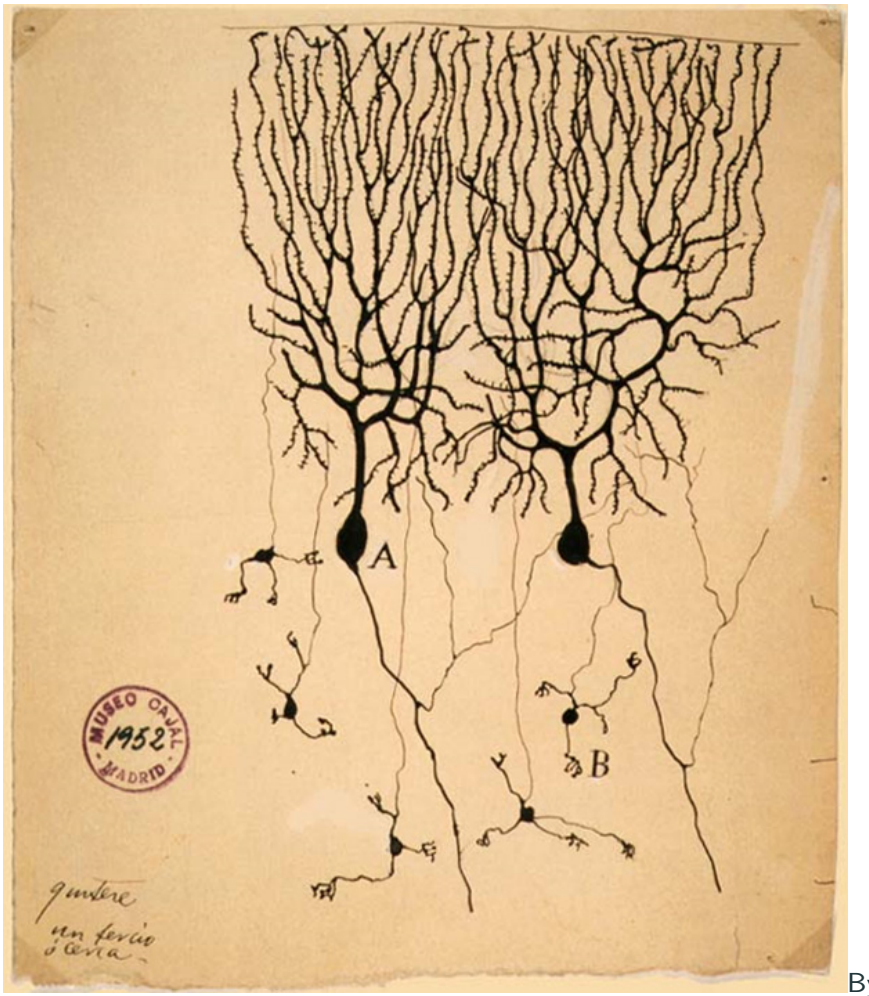
A learning algorithm $\hat{h}_N$ with $S = \{(x_i, y_i)\}_{i=1}^N$ where $(x_i, y_i) \sim \pi(x, y)$ of the form

$$\hat{h}_N \in \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i))$$

is called Empirical Risk-Minimizer.

Neural Networks

Motivation from biology



By Santiago Ramn y Cajal in 1899 see

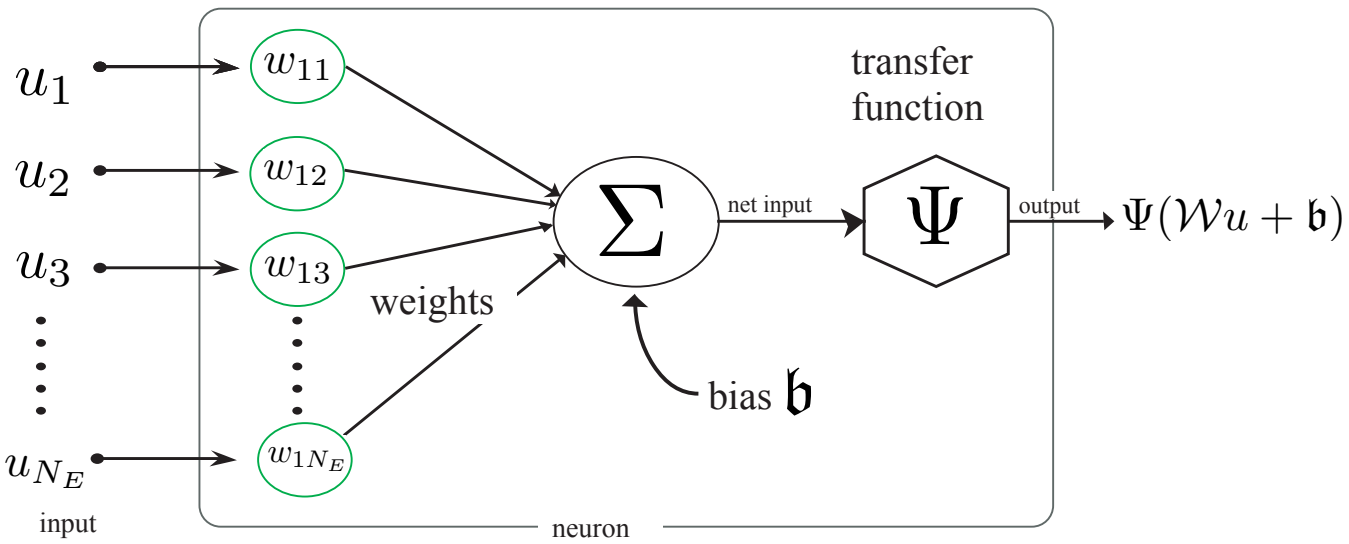
https://de.wikipedia.org/wiki/Santiago_Ram\protect\protect\begingroup\immediate\write\@unused\def\MessageBreakΩ\

let\protect\edefYouneedtoprovideadefinitionwith\DeclareInputText\MessageBreakor\DeclareInputMathbeforeusingthiskey.

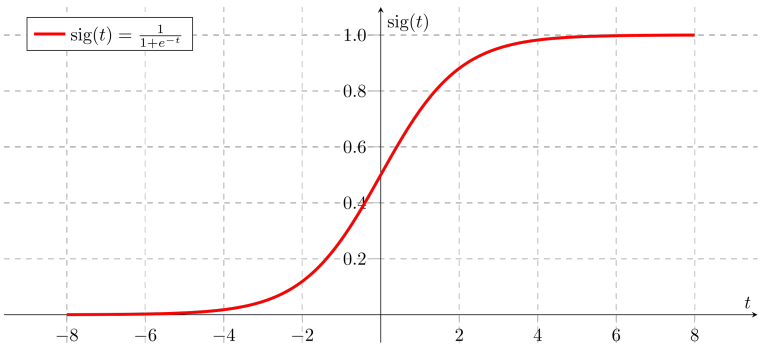
\errhelp\let\def\MessageBreakΩ(inputenc)\def\errmessagePackageinputencError:

Keyboardcharacterusedisundefined\MessageBreakininputencoding‘utf8’

Neuron


u_1
u_2
u_3
u_{N_E}
input
w_{11}
w_{12}
w_{13}
w_{1N_E}
weights
Σ
net input
transfer function
Ψ
output
$\Psi(\mathcal{W}u + \mathfrak{b})$
bias $\mathfrak{b}$
neuron

Activation function example: sigmoid



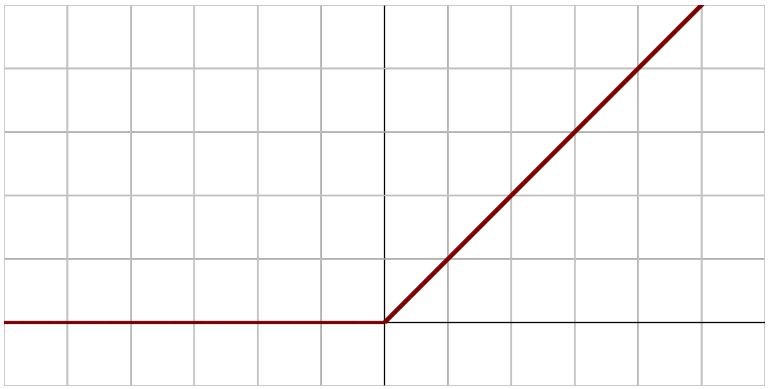
Sigmoid function:

$$\text{sig}(t) = \frac{1}{1+e^{-t}}$$

Properties:

- Derivative: $\dfrac{1 + e^{-x} + xe^{-x}}{(1 + e^{-x})^2}$
- $\text{sig}'(t) = \text{sig}(t)\,(1 - \text{sig}(t))$

Activation function example: ReLu



Rectified linear unit:

$$\text{ReLu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

$$= \max\{0, x\}$$

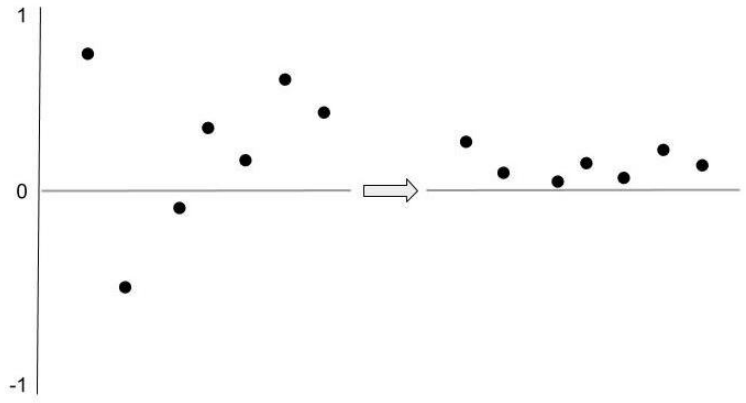
Properties:

- Derivative:

$$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases} \tag{3}$$

- very popular for Deep RL
- Dying ReLU problem - vanishing gradient problem.

Activation function example: Softmax



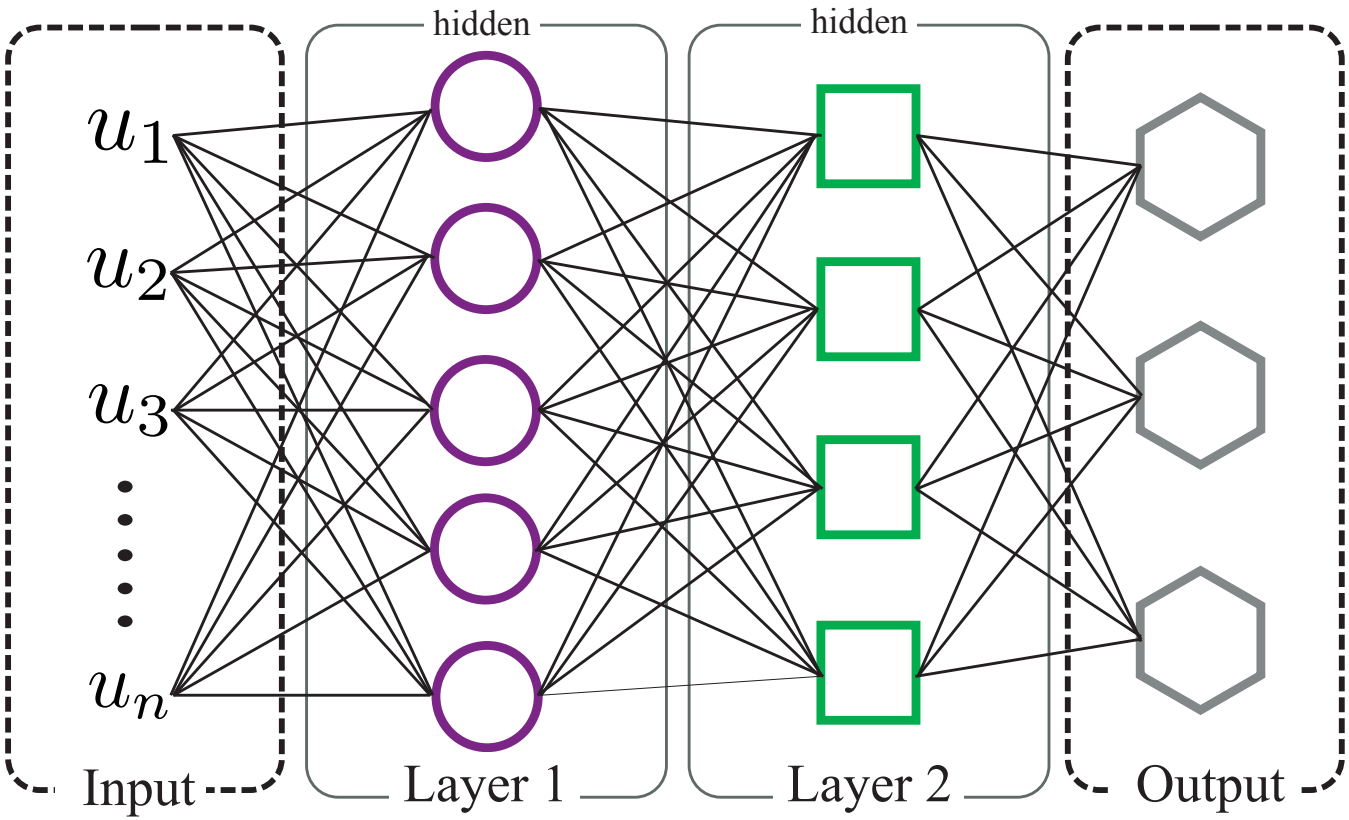
Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad \text{for } i = 1, \ldots, K$$
$$\text{and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$

Properties:

- Derivative:
$$\frac{\partial}{\partial q_k}\sigma(\mathbf{q}, i) = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k)). \tag{4}$$
- used in to normalize the output (map to a probability distribution)
- also used in RL to convert action values into action probabilities

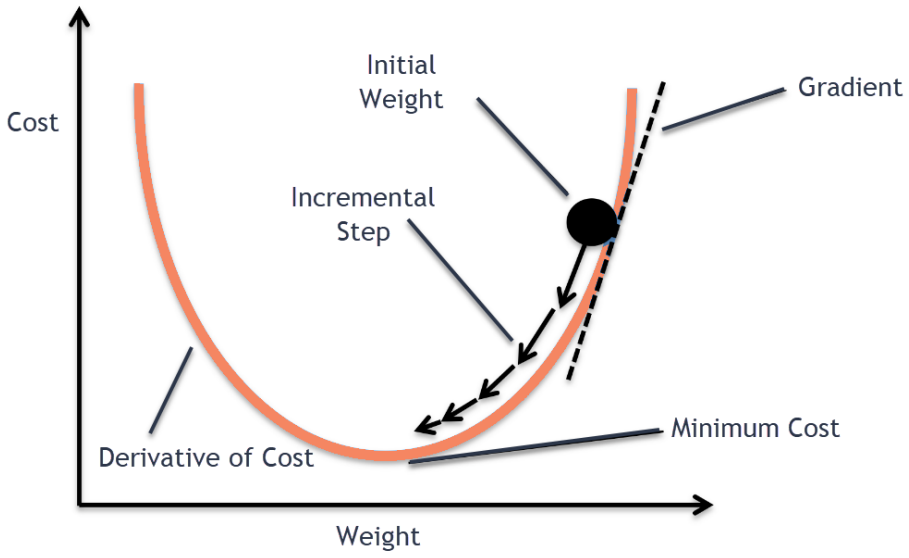
Multilayer perceptron



Training Neural Network

1. Choose network architecture:
   - activation functions
   - hidden layers (shallow or deep)
   - number of neurons
   - etc.
2. Choose appropriate loss function E, e.g., least squares
3. Find minima via:
   - stochastic gradient descent
   - Backpropagation

Stochastic Gradient Descent



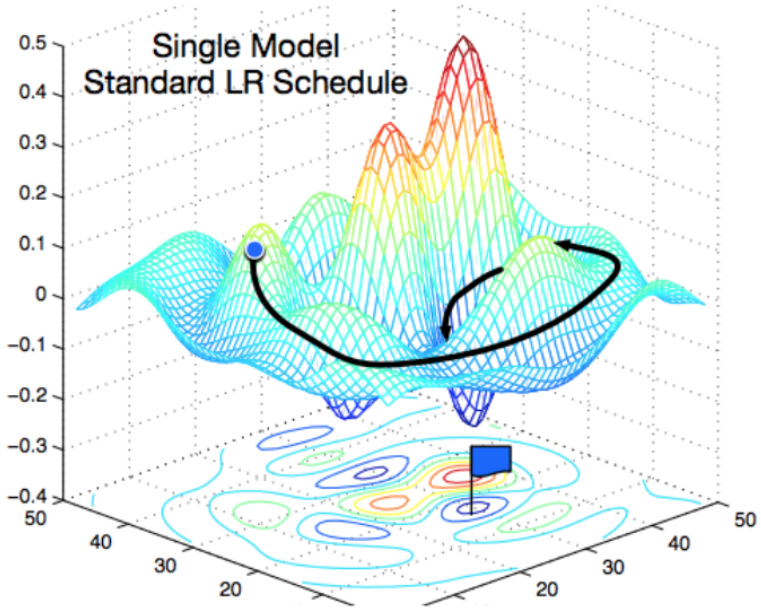


Image ref: https://morioh.com/p/bc6bc20e9739 and

https://medium.com/38th-street-studios/exploring-stochastic-gradient-descent-with-restarts-sgdr-fa206c38a74e

Iterative weight improvement:

$$w := w - \eta \nabla E_i(w). \tag{5}$$

Backpropagation

Backpropagation

Stochastic Approximation of a Mean

Definition: Let X a random variable bounded in $[0, 1]$ with mean $\mu = \mathbb{E}[X]$ and $x_n \sim X$ be n i.i.d. realizations of X. The stochastic approximation of the mean is,

$$\mu_n = (1 - \eta_n)\mu_{n-1} + \eta_n x_n \tag{6}$$

with $\mu_1 = x_1$ and where (η_n) is a sequence of learning steps.

Remark: When $\eta_n = \frac{1}{n}$ this is the recursive definition of empirical mean.

Stochastic Approximation of a Mean

Theorem: If for any n,$\eta_n \geq 0$ and are such that

$$\sum_{n\geq 0} \eta_n = \infty, \quad \sum_{n\geq 0} \eta_n^2 < \infty, \tag{7}$$

then

$$\mu_n \xrightarrow{a.s.} \mu \tag{8}$$

and we say that μ_n is a consistent estimator.

Stochastic Approximation of a Mean

Proof We focus on the case $\eta_n = n^{-\alpha}$. In order to satisfy the two conditions we need $1/2 < \alpha \leq 1$. For example for $\alpha = 1/2$ and $\alpha = 2$ we obtain:

$$\alpha = 2 \implies \sum_{n \geq 0} \frac{1}{n^2} = \frac{\pi^2}{6} < \infty \tag{9}$$

or

$$\alpha = 1/2 \implies \sum_{n \geq 0} (\frac{1}{\sqrt{n}})^2 = \sum_{n \geq 0} \frac{1}{n} = \infty \tag{10}$$

Stochastic Approximation of a Mean

Proof Case $\alpha = 1$: let $(\epsilon_k)_k$ be a sequence such that $\epsilon_k \to 0$, almost sure convergence corresponds to where (η_n) is a sequence of learning steps.

$$\mathbb{P}\Big(\lim_{n\to\infty} \mu_n = \mu\Big) = \mathbb{P}(\forall\ k \exists\ n_k\ \forall n \geq n_k\, |\mu_n - \mu| \leq \epsilon_k) = 1 \tag{11}$$

From Chernoff-Hoeffding inequality for any fixed n

$$\mathbb{P}\Big(|\mu_n - \mu| \geq \epsilon\Big) \leq 2\exp(-2n\epsilon^2) \tag{12}$$

Let $\{E_n\}$ be a sequence of events $E_n = \{|\mu_n - \mu| \geq \epsilon\}$. From Chernoff-Hoeffding

$$\sum_{n\geq 1} \mathbb{P}(E_n) < \infty \tag{13}$$

and from Borel-Cantelli lemma we obtain that with probability 1 there exist only a finite number of n values such that $|\mu_n - \mu| \geq \epsilon$.

Proof: Stochastic Approximation of a Mean

Then for any ϵ_k there exist only a finite number of instants were $|\mu_n - \mu| \geq \epsilon_k$ which corresponds to have $\exists n_k$ such that

$$\mathbb{P}(\forall n \geq n_k, |\mu_n - \mu| \leq \epsilon_k) = 1 \tag{14}$$

Repeating for all ϵ_k in the sequence leads to the statement.

Remark: when $\alpha = 1$, μ_n is the Monte-Carlo estimate and this corresponds to the strong law of large numbers. A more precise and accurate proof is here: `http://terrytao.wordpress.com/2008/06/18/the-strong-law-of-large-numbers/`

Proof: Stochastic Approximation of a Mean

case: $1/2 < \alpha < 1$. The stochastic approximation μ_n is

$$\mu_1 = x_1 \tag{15}$$

$$\mu_2 = (1-\eta_2)\mu_1 + \eta_2 x_2 = (1-\eta_2)x_1 + \eta_2 x_2 \tag{16}$$

$$\mu_3 = (1-\eta_3)\mu_2 + \eta_3 x_3 = (1-\eta_2)(1-\eta_3)x_1 + \eta_2(1-\eta_3)x_2 + \eta_3 x_3 \tag{17}$$

$$\dots \tag{18}$$

$$\mu_n = \sum_{i=1}^{n} \lambda_i x_i \tag{19}$$

with $\lambda_i = \eta_i \prod_{j=i+1}^{n}(1-\eta_j)$ such that $\sum_{i=1}^{n} \lambda_i = 1$. By C-H inequality

$$\mathbb{P}(|\sum_{i=1}^{n} \lambda_i x_i - \sum_{i=1}^{n} \lambda_i \mathbb{E}[x_i]| \geq \epsilon) = \mathbb{P}(|\mu_n - \mu| \geq \epsilon) \leq e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n} \lambda_i^2}} \tag{20}$$

Proof: Stochastic Approximation of a Mean

case: $1/2 < \alpha < 1$. From the definition of λ_i

$$\log(\lambda_i) = \log(\eta_i) + \sum_{j=i+1}^{n} \log(1-\eta_j) \leq \log(\eta_i) - \sum_{j=i+1}^{n} \eta_j \tag{21}$$

since $\log(1-x) < -x$. Thus $\lambda_i \leq \eta_i e^{-\sum_{j=i+1}^{n} \eta_j}$ and for any $1 \leq m \leq n$,

$$\sum_{i=1}^{n} \lambda_i^2 \leq \sum_{i=1}^{n} \eta_i^2 e^{-2\sum_{j=i+1}^{n} \eta_j} \tag{22}$$

$$\leq \sum_{i=1}^{m} e^{-2\sum_{j=i+1}^{n} \eta_j} + \sum_{i=m+1}^{n} \eta_i^2 \tag{23}$$

$$\leq m e^{-2(n-m)\eta_n} + (n-m)\eta_m^2 \tag{24}$$

$$= m e^{-2(n-m)n^{-\alpha}} + (n-m)m^{-2\alpha} \tag{25}$$

$$\tag{26}$$

Proof: Stochastic Approximation of a Mean

case: $1/2 < \alpha < 1$. Let $m = n^\beta$ with

$$\beta = (1 + \alpha/2)/2$$

(i.e., $1 - 2\alpha\beta = 1/2 - \alpha$)

$$\sum_{i=1}^{n} \lambda_i^2 \leq n e^{-2(1-n^{-1/4})n^{1-\alpha}} + n^{1/2-\alpha} \leq 2n^{1/2-\alpha} \tag{27}$$

for n big enough, which leads to

$$\mathbb{P}(|\mu - n - \mu| \geq \epsilon) \leq e^{-\frac{\epsilon^2}{n^{1/2-\alpha}}} \tag{28}$$

From this point we follow the same steps as for $\alpha = 1$ (application of the Borel-Cantelli lemma) and obtain the convergence result for μ_n

Robbins-Monro (1951) algorithm

Theorem: Given a noisy function f , find $x^\star$ such that $f(x^\star) = 0$. In each x_n, observe $y_n = f(x_n) + b_n$ (with b_n a zero-mean independent noise) and compute

$$x_{n+1} = x_n - \eta_n y_n \tag{29}$$

If f is an increasing function, then under the same assumptions on the learning step

$$x_n \xrightarrow{a.s.} x^\star \tag{30}$$

Kiefer-Wolfowitz (1952) algorithm

Theorem: Given a function f and noisy observations of its gradient, find $x^\star = \arg\min f(x)$.. In each x_n, observe $g_n = \nabla f(x_n) + b_n$ (with b_n a zero-mean independent noise) and compute

$$x_{n+1} = x_n - \eta_n g_n \tag{31}$$

If the Hessian $\nabla^2 f$ is positive, then under the same assumptions on the learning step

$$x_n \xrightarrow{a.s.} x^\star \tag{32}$$

Remark: this is often referred to as the stochastic gradient algorithm.