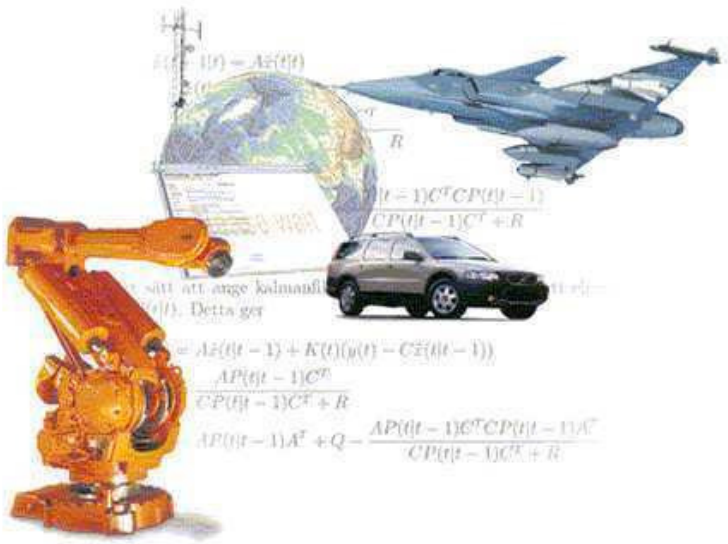
The Origins of Machine Learning in Control - Stochastic Approximation



Lennart Ljung

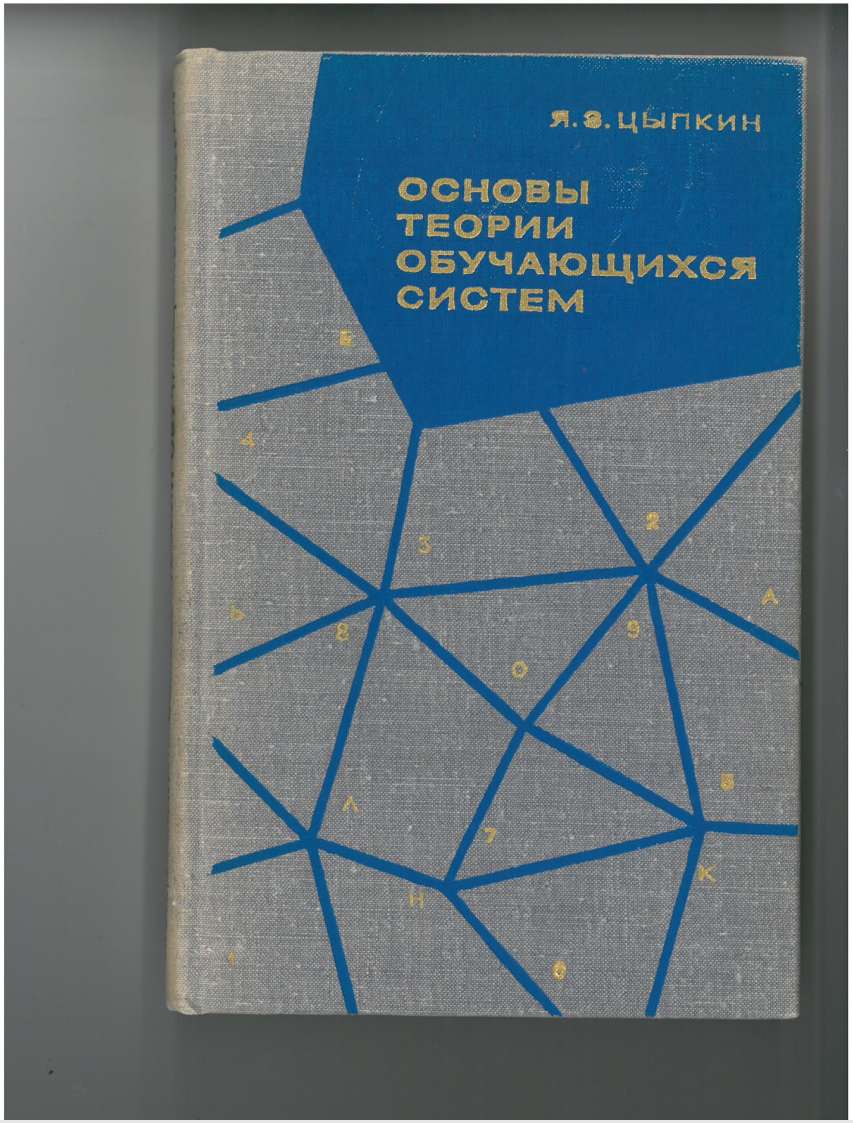
Automatic Control, ISY, Linköpings Universitet

The Community in the Late 60's, Early 70's

When I started as PhD student in 1970, the community was bubbling with learning concepts like

- *Learning Machines*, Nils Nilsson, 1965
- *Foundations of the Theory of Learning Systems*, Tsypkin, 1970
- *Adaptation and Learning in Automatic Systems*, Tsypkin, 1971
- ....

Ya. Z. Tsypkin



Tsypkin was one of the leaders in this trend on learning systems, and I (with K.J. Åström) decided that I would spend 6 months in Moscow with Tsypkin's group in 1972 as a "pre-doc".

In those days, “learning systems” was really another term for applying *Stochastic Approximation* to various control and estimation criteria.

In those days, “learning systems” was really another term for applying *Stochastic Approximation* to various control and estimation criteria.

Annals of Mathematical Statistics, Vol 22 1951, pp 400-407

A STOCHASTIC APPROXIMATION METHOD[1]

By Herbert Robbins and Sutton Monro

University of North Carolina

1. Summary. Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making

In those days, “learning systems” was really another term for applying *Stochastic Approximation* to various control and estimation criteria.

Annals of Mathematical Statistics, Vol 22 1951, pp 400-407

A STOCHASTIC APPROXIMATION METHOD[1]

By Herbert Robbins and Sutton Monro

University of North Carolina

1. Summary. Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making

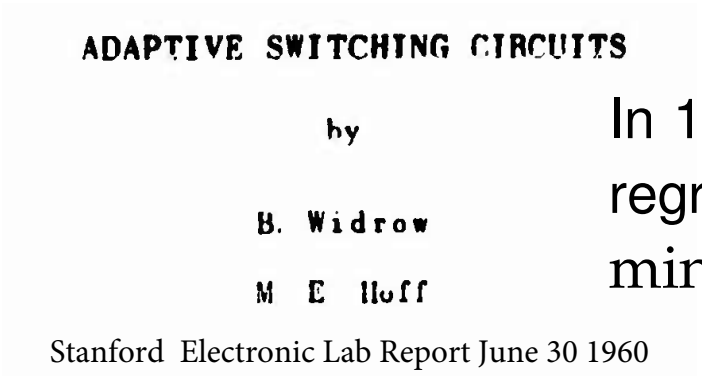
The Robbins-Monro scheme, 1951

Solve $M(x) = 0$, where measurements $H(x,t) = M(x) + e(t)$ are available:

Do $x_t = x_{t-1} + \gamma_t H(x_t, t)$, with $\sum \gamma_t = \infty \quad \sum \gamma_t^2 < \infty$

Then $x_t \to x^*, \quad M(x^*) = 0$

LMS - Another Thread


ADAPTIVE SWITCHING CIRCUITS

by

B. Widrow

M E Hoff

Stanford Electronic Lab Report June 30 1960

In 1960 Bernie Widrow studied the linear regression problem $y(t) = \varphi^T(t)\theta + e(t)$

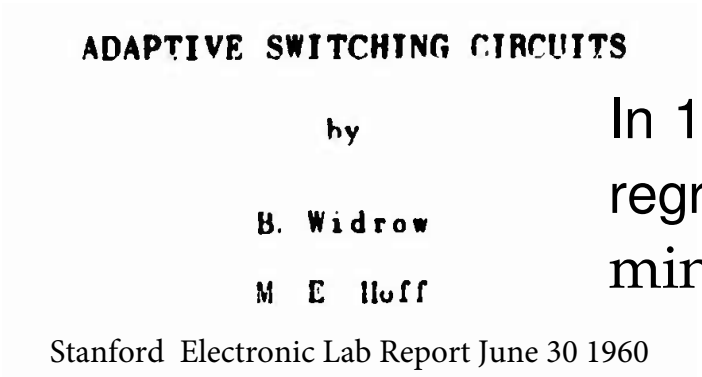
$$\min V(\theta) = \min E(y(t) - \varphi^T(t)\theta)^2$$

He suggested to solved it by the Least Mean Squares (LMS) algorithm

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \hat{y}(t))$$

where $\hat{y}(t)$ is the model output at time t.

LMS - Another Thread


ADAPTIVE SWITCHING CIRCUITS

by

B. Widrow

M E Hoff

Stanford Electronic Lab Report June 30 1960

In 1960 Bernie Widrow studied the linear regression problem $y(t) = \varphi^T(t)\theta + e(t)$

$$\min V(\theta) = \min E(y(t) - \varphi^T(t)\theta)^2$$

He suggested to solved it by the Least Mean Squares (LMS) algorithm

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \hat{y}(t))$$

where $\hat{y}(t)$ is the model output at time t. A moment's reflection shows that this is the RM scheme applied to

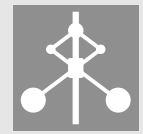
$V'(\theta) = E\varphi(t)(y(t) - \varphi^T(t)\theta)$ if $\hat{y}(t) = \varphi^T(t)\hat{\theta}(t-1)$

But Widrow introduced an anthropomorphic language around this with “learning”, “teaching”, “training phase”:

SUMMARY

Adaptive or "learning" systems can automatically modify their own structures to optimize performance based on past experiences. The system designer "teaches" by showing the system examples of input signals or patterns and simultaneously what he would like the output to be for each input. The system in turn organizes itself to comply as well as possible with the wishes of the designer.

An adaptive pattern classification machine (called "Adaline", for adaptive linear) has been devised to illustrate adaptive behavior and artificial learning. During a training phase, crude geometric patterns



Widrow preferred let the model output to be quantized:

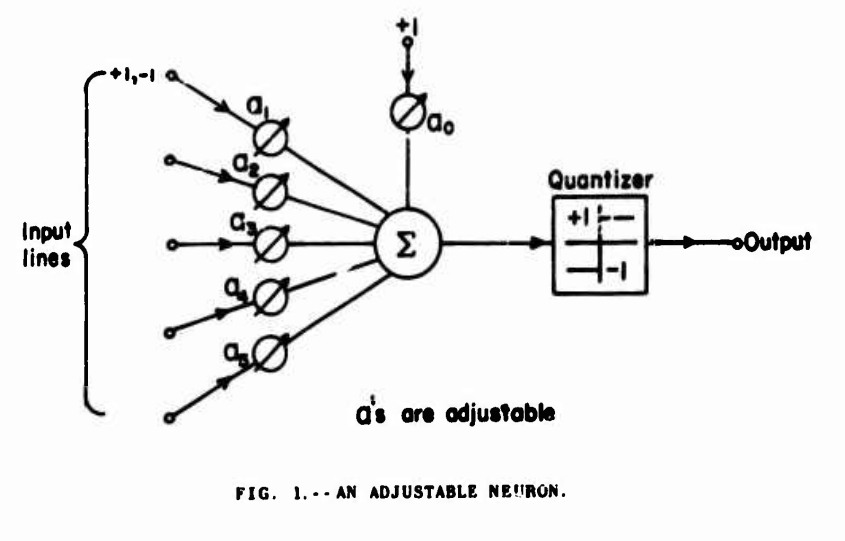


FIG. 1.--AN ADJUSTABLE NEURON.

and called the element a “neuron” (inspired by von Neuman, 1956).

Widrow preferred let the model output to be quantized:

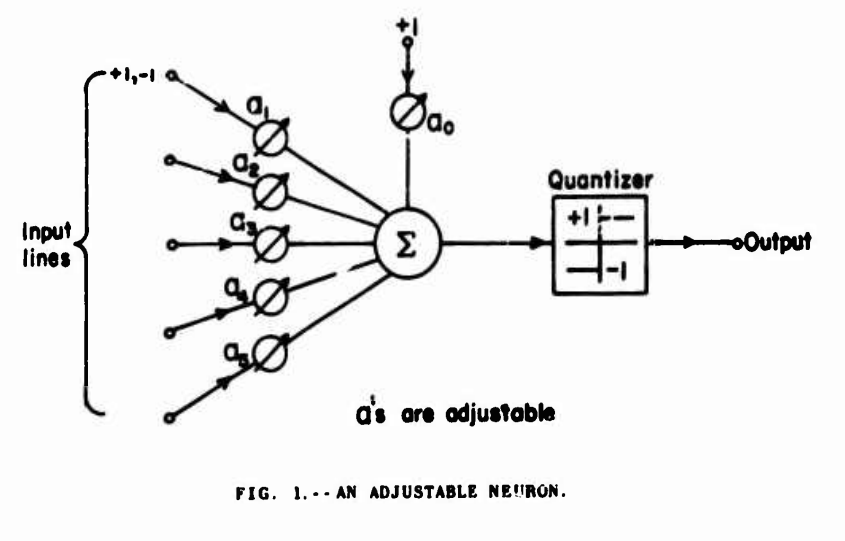


FIG. 1.--AN ADJUSTABLE NEURON.

and called the element a “neuron” (inspired by von Neuman, 1956). This links to the concept on *perceptrons* introduced by Rosenblatt, Cornell, 1957.

Widrow preferred let the model output to be quantized:

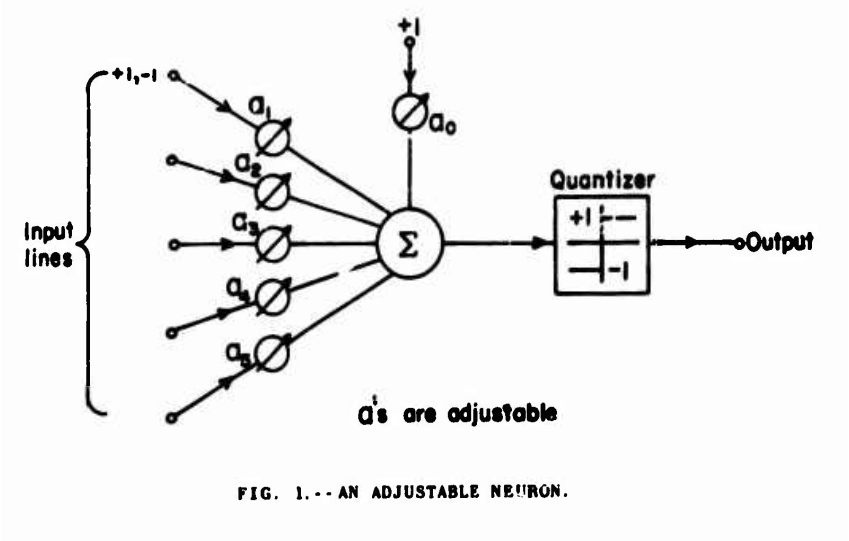


FIG. 1.--AN ADJUSTABLE NEURON.

and called the element a “neuron” (inspired by von Neuman, 1956). This links to the concept on *perceptrons* introduced by Rosenblatt, Cornell, 1957.

We recognize the figure above as a “one hidden layer neural network” (with one neuron).

Back to SA and Recursive Identification

The LMS or Robbins-Monro scheme for estimating θ in a linear regression

$$y(t) = \varphi^T(t)\theta + e(t);\ \hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

is an example of a *recursive identification algorithm*, to build models of dynamical systems, on-line, without storing all observations over time t.

Back to SA and Recursive Identification

The LMS or Robbins-Monro scheme for estimating θ in a linear regression

$$y(t) = \varphi^T(t)\theta + e(t);\ \hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

is an example of a *recursive identification algorithm*, to build models of dynamical systems, on-line, without storing all observations over time t.

In this linear regression case, we can write the basic Least Squares estimate

$$\hat{\theta}(t) = [\sum_{k=1}^{t} \varphi(k)\varphi^T(k)]^{-1} \sum \varphi(k)y(k) = R^{-1}(t) \sum \varphi(k)y(k)$$

Back to SA and Recursive Identification

The LMS or Robbins-Monro scheme for estimating θ in a linear regression

$$y(t) = \varphi^T(t)\theta + e(t);\ \hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

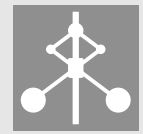
is an example of a *recursive identification algorithm*, to build models of dynamical systems, on-line, without storing all observations over time t.

In this linear regression case, we can write the basic Least Squares estimate

$$\hat{\theta}(t) = [\sum_{k=1}^{t} \varphi(k)\varphi^T(k)]^{-1} \sum \varphi(k)y(k) = R^{-1}(t) \sum \varphi(k)y(k)$$

which can be exactly rewritten recursively as (RLS)

$$\hat{\theta}(t) = \hat{\theta}(t-1) + R^{-1}(t)\varphi(t)(y(t) - \varphi^T(t)\theta(t-1))$$



Note that this is exactly LMS,

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

only that γ is replaced with R^{-1}, the inverse of the Hessian of the criterion, changing the gradient step to a Newton step.

Note that this is exactly LMS,

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

only that γ is replaced with R^{-1}, the inverse of the Hessian of the criterion, changing the gradient step to a Newton step.
This indicates the links between SA and recursive identification also for more sophisticated model structures than linear regressions.

Note that this is exactly LMS,

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma\varphi(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

only that γ is replaced with R^{-1}, the inverse of the Hessian of the criterion, changing the gradient step to a Newton step.
This indicates the links between SA and recursive identification also for more sophisticated model structures than linear regressions.
And Recursive identification is the essential engine behind most adaptive ("learning") algorithms in control and signal processing.

Convergence of Learning Algorithms

Inspired from the Robbins- Monro scheme, a (recursive) learning algorithm has the structure

$$x(t) = x(t-1) + \gamma(t)Q(x(t-1), e(t)); \quad \gamma(t) \to 0, \quad \sum \gamma(t) = \infty$$

x is the "decision variable" , e is some noise term and Q is a measurement of the current behaviour. How does $x(t)$ behave? Does it converge to something as $t \to \infty$?

Inspired from the Robbins- Monro scheme, a (recursive) learning algorithm has the structure

$$x(t) = x(t-1) + \gamma(t)Q(x(t-1), e(t)); \quad \gamma(t) \to 0, \quad \sum \gamma(t) = \infty$$

x is the "decision variable" , e is some noise term and Q is a measurement of the current behaviour. How does $x(t)$ behave? Does it converge to something as $t \to \infty$?
Define the ODE

$$\frac{d}{d\tau}x(\tau) = M(x(\tau)); \quad M(x) = EQ(x, e)$$

Convergence of Learning Algorithms

Inspired from the Robbins- Monro scheme, a (recursive) learning algorithm has the structure

$$x(t) = x(t-1) + \gamma(t)Q(x(t-1), e(t)); \quad \gamma(t) \to 0, \quad \sum \gamma(t) = \infty$$

x is the "decision variable" , e is some noise term and Q is a measurement of the current behaviour. How does $x(t)$ behave? Does it converge to something as $t \to \infty$?
Define the ODE

$$\frac{d}{d\tau}x(\tau) = M(x(\tau)); \quad M(x) = EQ(x, e)$$

Then the paths of $x(t)$ asymptotically follow the trajectories of $x(\tau)$.
Stability of the ODE implies convergence of the learning algorithm.

Connections to Machine Learning

Machine Learning is in the end, I would say, about building a function

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t.

Connections to Machine Learning

Machine Learning is in the end, I would say, about building a function

$$y = f(x)$$

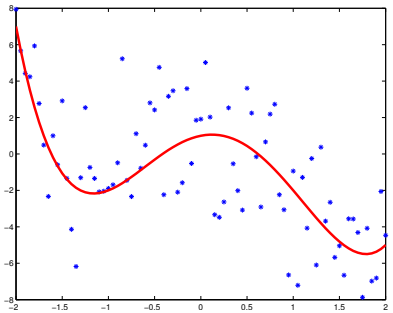
from observations of stimuli x_t and corresponding (noisy) responses y_t. The data may come from anywhere. The spaces where x and y live can be quite diverse (from high-school curve fitting,

Connections to Machine Learning

Machine Learning is in the end, I would say, about building a function

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t. The data may come from anywhere. The spaces where x and y live can be quite diverse (from high-school curve fitting,

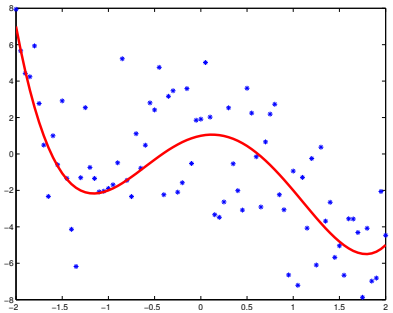


Connections to Machine Learning

Machine Learning is in the end, I would say, about building a function

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t. The data may come from anywhere. The spaces where x and y live can be quite diverse (from high-school curve fitting,



to classification of Facebook texts)

Parameterizations

We typically need a way to parameterize the function f. One way is to select *features* $\Phi(x) \in R^p$ by some *projections*. The selection may itself be parameterized: $\Phi(x, \eta)$. f will now be a function of $\Phi(x, \eta))$ parameterized by θ; $f(\theta, \Phi(x, \eta))$.

Parameterizations

We typically need a way to parameterize the function f. One way is to select *features* $\Phi(x) \in R^p$ by some *projections*. The selection may itself be parameterized: $\Phi(x, \eta)$. f will now be a function of $\Phi(x, \eta))$ parameterized by θ; $f(\theta, \Phi(x, \eta))$.
Finding an informative projection Φ is a crucial problem and is related to "deep learning"

Parameterizations

We typically need a way to parameterize the function f. One way is to select *features* $\Phi(x) \in R^p$ by some *projections*. The selection may itself be parameterized: $\Phi(x, \eta)$. f will now be a function of $\Phi(x, \eta))$ parameterized by θ; $f(\theta, \Phi(x, \eta))$.
Finding an informative projection Φ is a crucial problem and is related to "deep learning"
For simplicity we ignore η and let f be a linear function of θ:

$$f(x) = \Phi^T(x)\theta$$

and measure the fit between y and f by a quadratic criterion $\|y - \Phi^T(x)\theta)\|^2$ (Possibly with an additive regularizor $\Omega(\theta)$).

Parameterizations

We typically need a way to parameterize the function f. One way is to select *features* $\Phi(x) \in R^p$ by some *projections*. The selection may itself be parameterized: $\Phi(x, \eta)$. f will now be a function of $\Phi(x, \eta))$ parameterized by θ; $f(\theta, \Phi(x, \eta))$.
Finding an informative projection Φ is a crucial problem and is related to "deep learning"
For simplicity we ignore η and let f be a linear function of θ:

$$f(x) = \Phi^T(x)\theta$$

and measure the fit between y and f by a quadratic criterion $\|y - \Phi^T(x)\theta)\|^2$ (Possibly with an additive regularizor $\Omega(\theta)$).
which make the estimation problem in this special case a *linear least squares* problem.

Convergence Rates

Then we can apply the Robbins-Monro (stochastic gradient, LMS) method

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma_t \Phi(x_t)[y(t) - \Phi^T(x_t)\hat{\theta}(t-1)]$$

or the stochastic Newton (RLS) method

$$\hat{\theta}(t) = \hat{\theta}(t-1) + R(t)^{-1}\Phi(x_t)[y(t) - \Phi^T(x_t)\hat{\theta}(t-1)];\ R(t) \sim \sum \Phi\Phi^T$$

Rates: Importance of the Conditioning of R

The convergence rate of RLS is always $\sim \frac{1}{t}$:

$$\lim_{t\to\infty} tE\|\hat{\theta}(t) - \theta_0\|^2 = \Pi; \quad \text{Cramér-Rao}$$

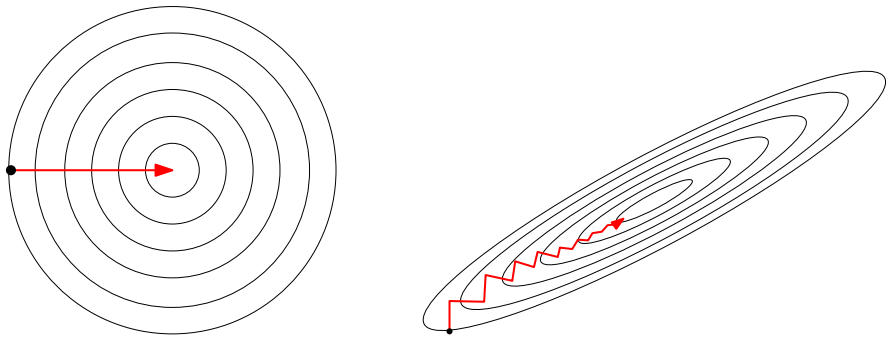
[1] UCI repository of Machine learning databases

Rates: Importance of the Conditioning of R

The convergence rate of RLS is always $\sim \frac{1}{t}$:

$$\lim_{t\to\infty} tE\|\hat{\theta}(t) - \theta_0\|^2 = \Pi; \quad \text{Cramér-Rao}$$

while the convergence rate of LMS can be very slow if R is ill-conditioned.



So aim for using RLS.

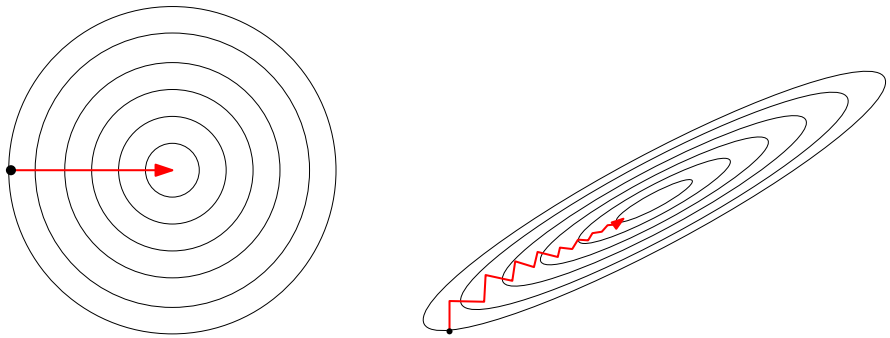
[1] UCI repository of Machine learning databases

Rates: Importance of the Conditioning of R

The convergence rate of RLS is always $\sim \frac{1}{t}$:

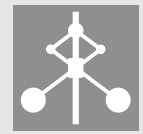
$$\lim_{t\to\infty} tE\|\hat{\theta}(t) - \theta_0\|^2 = \Pi; \quad \text{Cramér-Rao}$$

while the convergence rate of LMS can be very slow if R is ill-conditioned.



So aim for using RLS. But R is $p \times p$, p being the number of features

[1] UCI repository of Machine learning databases

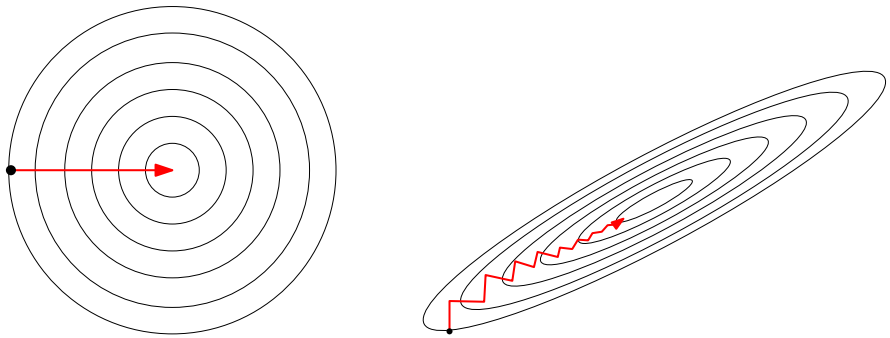


Rates: Importance of the Conditioning of R

The convergence rate of RLS is always $\sim \frac{1}{t}$:

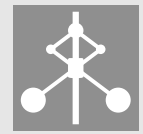
$$\lim_{t\to\infty} tE\|\hat{\theta}(t) - \theta_0\|^2 = \Pi; \quad \text{Cramér-Rao}$$

while the convergence rate of LMS can be very slow if R is ill-conditioned.



So aim for using RLS. But R is $p \times p$, p being the number of features - which could be very large (~ 1300000 in "big data" machine learning like the "News" benchmark[1]) and the Newton step is then forbidding.

[1] UCI repository of Machine learning databases



An Interesting Observation: Accelerated Convergence Polyak and Juditsky, 1992

Define “a second round of averaging”:

$$\tilde{\theta}(t) = \tilde{\theta}(t-1) + \tilde{\gamma}_t \Phi(x_f)[y(t) - \Phi^T(x_t)\tilde{\theta}(t-1)]$$

$$\hat{\theta}(t) = \frac{1}{t}\sum \tilde{\theta}(k)$$

$\tilde{\gamma}$ "slowly decaying, or even a small constant.

An Interesting Observation: Accelerated Convergence Polyak and Juditsky, 1992

Define “a second round of averaging”:

$$\tilde{\theta}(t) = \tilde{\theta}(t-1) + \tilde{\gamma}_t \Phi(x_f)[y(t) - \Phi^T(x_t)\tilde{\theta}(t-1)]$$

$$\hat{\theta}(t) = \frac{1}{t}\sum \tilde{\theta}(k)$$

$\tilde{\gamma}$ "slowly decaying, or even a small constant.Then $\hat{\theta}(t)$ has the same convergence rate as the Newton estimate!

$$\lim_{t\to\infty} tE\|\hat{\theta}(t) - \theta_0\|^2 = \Pi; \quad \text{Cramér-Rao}$$

No $p \times p$ matrix ever formed!

A Digression, "Self-learning"

In the setup

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t it could be that the values y are never measured, but still f is sought. (“unsupervised learning, learning without a teacher, self-learning”).

A Digression, "Self-learning"

In the setup

$$y = f(x)$$

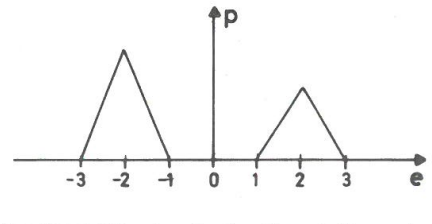
from observations of stimuli x_t and corresponding (noisy) responses y_t it could be that the values y are never measured, but still f is sought. (“unsupervised learning, learning without a teacher, self-learning”). Is that at all possible?

A Digression, "Self-learning"

In the setup

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t it could be that the values y are never measured, but still f is sought. (“unsupervised learning, learning without a teacher, self-learning”). Is that at all possible? Well, suppose we have a classification problem, so f is either A or B and x is distributed according to



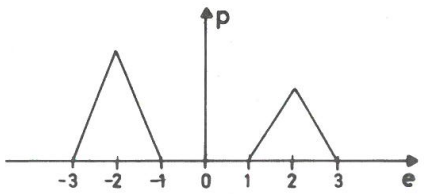
So it is clear that it can be done,but can it be done recursively?

A Digression, "Self-learning"

In the setup

$$y = f(x)$$

from observations of stimuli x_t and corresponding (noisy) responses y_t it could be that the values y are never measured, but still f is sought. (“unsupervised learning, learning without a teacher, self-learning”). Is that at all possible? Well, suppose we have a classification problem, so f is either A or B and x is distributed according to



So it is clear that it can be done,but can it be done recursively?

The means of the distributions are -2 and 2. Suppose p% falls in the left bin.

Convergence Analysis

For the classification problem, let the algorithm be that $x_t < c_t$ is classified as A and B else. Let c_t be $(c_t^A + c_t^B)/2$ where c_t^A is the mean of the x:s classified as A so far and analogously for c_t^B. Then what happens?
The algorithm can be written as

$$c(t) = c(t-1) + \gamma(t)Q(c(t-1), x_t); \quad c(t) = \begin{bmatrix} c_t^A \\ c_t^B \end{bmatrix}$$

where Q is formed from the rules expressed above.

Convergence Analysis

For the classification problem, let the algorithm be that $x_t < c_t$ is classified as A and B else. Let c_t be $(c_t^A + c_t^B)/2$ where c_t^A is the mean of the x:s classified as A so far and analogously for c_t^B. Then what happens?
The algorithm can be written as

$$c(t) = c(t-1) + \gamma(t)Q(c(t-1), x_t); \quad c(t) = \begin{bmatrix} c_t^A \\ c_t^B \end{bmatrix}$$

where Q is formed from the rules expressed above. Will this simple recursive classifier always converge, regardless of starting values $x(0)$ and relative distribution p?

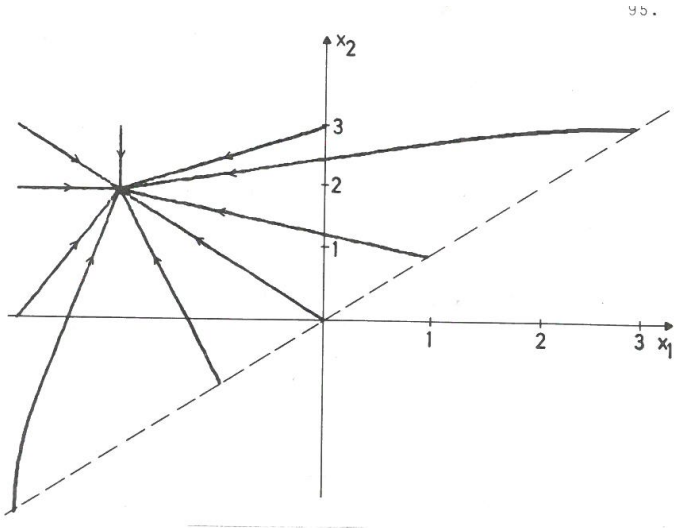
Convergence Analysis for the Unsupervised Classification

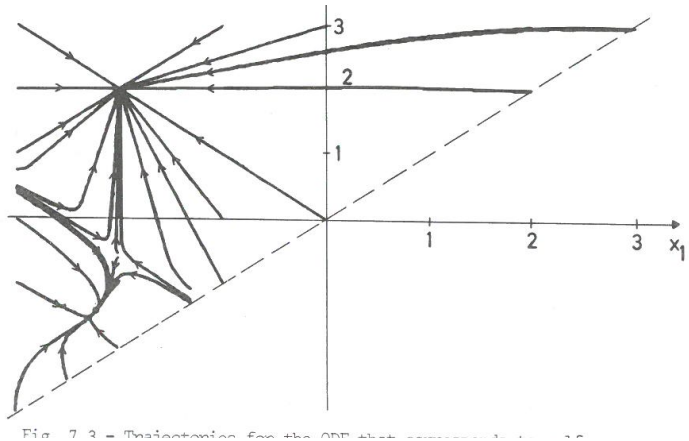
A convergence analysis can be based on the ODE's mentioned earlier, that correspond to infinitesimally small steps in the recursive algorithm:

Convergence Analysis for the Unsupervised Classification

A convergence analysis can be based on the ODE's mentioned earlier, that correspond to infinitesimally small steps in the recursive algorithm: Define the ODE

$$\frac{d}{d\tau}x(\tau) = M(x(\tau)); \quad M(x) = EQ(x,e)$$

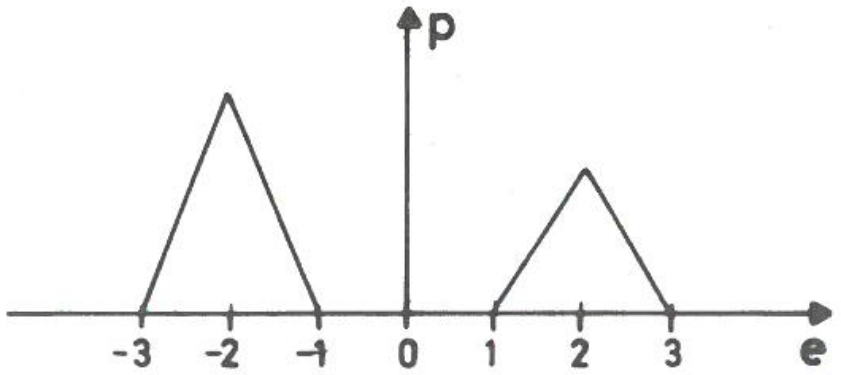




1) $p = 50\%$

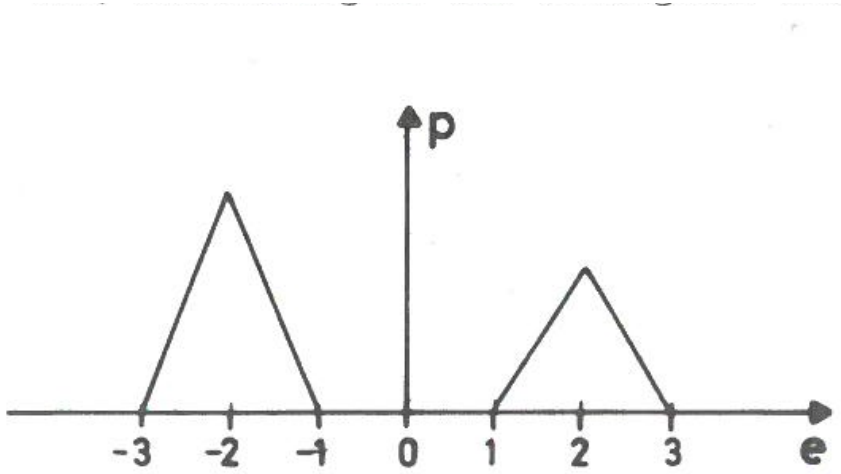
2) $p = 99\%$

Self-learning



It is clear that self-learning something about f in $y = f(x)$ from x only could be a meaningful problem.

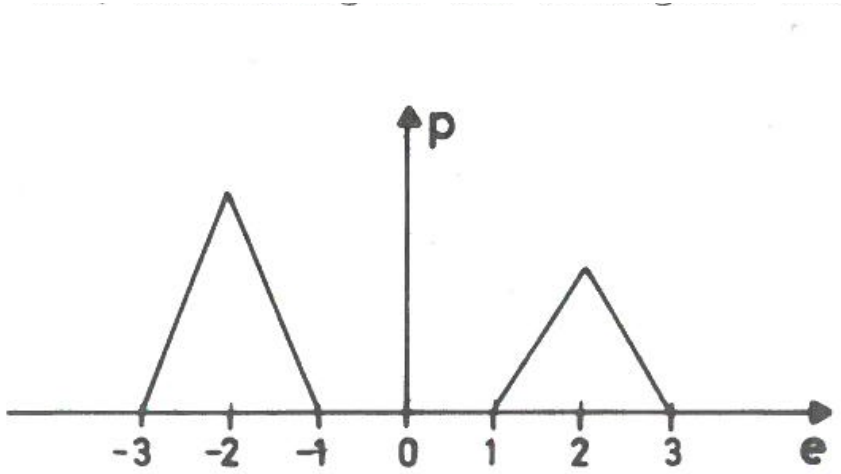
Self-learning



It is clear that self-learning something about f in $y = f(x)$ from x only could be a meaningful problem.

- Self-Organizing maps (Kohonen)

Self-learning

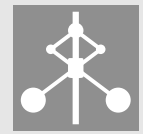


It is clear that self-learning something about f in $y = f(x)$ from x only could be a meaningful problem.

- Self-Organizing maps (Kohonen)
- Manifold learning

SA: Then and Now!

The stochastic approximation algorithms were very hot in the 70's.



SA: Then and Now!

The stochastic approximation algorithms were very hot in the 70's. But still today - 50 years later - stochastic approximation ideas are very central in (parts of) Machine Learning.

SA: Then and Now!

The stochastic approximation algorithms were very hot in the 70's.
But still today - 50 years later - stochastic approximation ideas are very central in (parts of) Machine Learning.
Invited keynote presentation at ERNSI in October:

- Francis Bach (INRIA): Beyond Stochastic Gradient Descent for Large-Scale Machine Learning

centered around Gradient and Newton algorithms, and accelerated convergence aspects for parameter estimation (like I did here).