ECE656-Machine Learning and Adaptive Systems Lectures 3 & 4

M.R. Azimi, Professor

Department of Electrical and Computer Engineering
Colorado State University

Fall 2015

What is Learning?

General Definition of Learning:

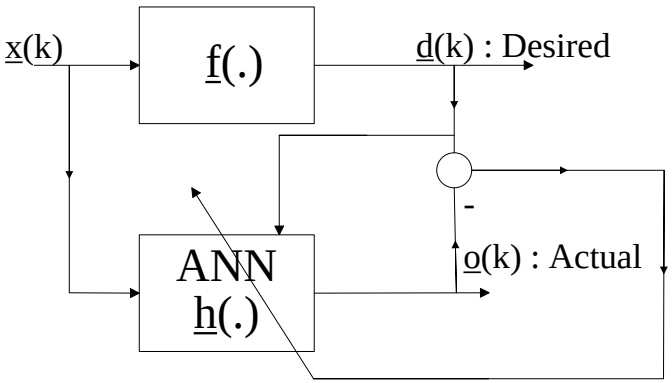
Any change in the behavior or performance brought about by the experience or contact with the environment.

Learning in humans is an inferred process, i.e. we cannot see it happening directly but observe the resultant changes in behavior. Learning in ANN is a more direct process and we can capture each learning step in a *cause-effect* relationship.

Designing a machine learning algorithm is based upon learning (supervised) a relationship that transforms inputs to outputs from a set of input-output examples drawn from Experience E.

Given a training set consisting of pairs of inputs-outputs $\{\underline{x}(k), \underline{d}(k)\}_{k=1}^{K}$ from E.
$\underline{x}(k)$: k^{th} training sample (pattern) vector of dimension $N \times 1$.
$\underline{d}(k)$: desired output (response) for input pattern $\underline{x}(k)$ of dimension $M \times 1$ where typically $M \leq N$.

The problem is viewed as a task T of approximating a continuous multi-variate mapping function $\underline{d}(k) = \underline{f}(\underline{x}(k))$ by a function $\underline{h}(\underline{w}, \underline{x}(k))$ as closely as possible (measured by some performance metric $\rho(.)$ wrt E) with $\underline{w}$ being the parameter vector.

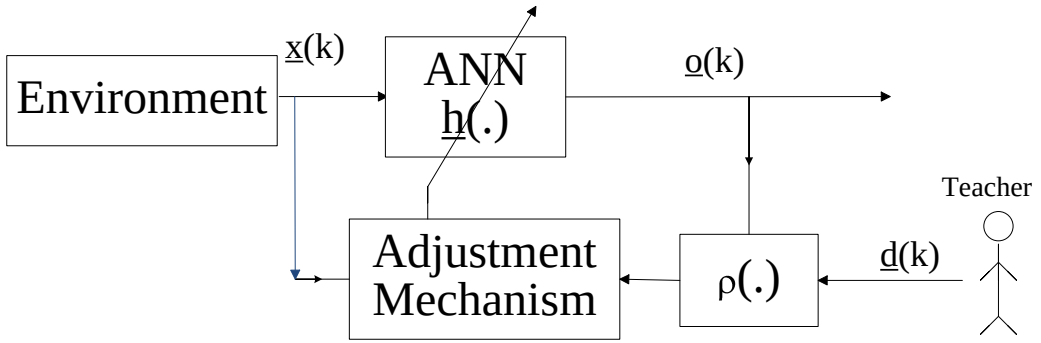


If we define a performance measure $\rho(\underline{h}(\underline{w}, \underline{x}(k)), \underline{f}(\underline{x}(k)))$, then for the optimal parameter vector $\underline{w}^*$,
$\rho(\underline{h}(\underline{w}^*, \underline{x}(k)), \underline{f}(\underline{x}(k)))) \leq \rho(\underline{h}(\underline{w}, \underline{x}(k)), \underline{f}(\underline{x}(k)))), \quad \forall k \in [1, K]$
where $\underline{w}$ is any other vector.

Note: Such learning rules are referred to as *Performance Learning*.
Examples of $\rho(.)$ are MSE, classification error, mutual information, etc.

1. Supervised Learning

- Requires an "external teacher".
- Teacher provides training sample pairs i.e. $\underline{x}(k)$ and $\underline{d}(k)$.
- Training is based upon *error learning*.
- Training is typically iterative.
- More accurate than unsupervised ones.
- Used for detection/classification, prediction/regression, and function approximation.

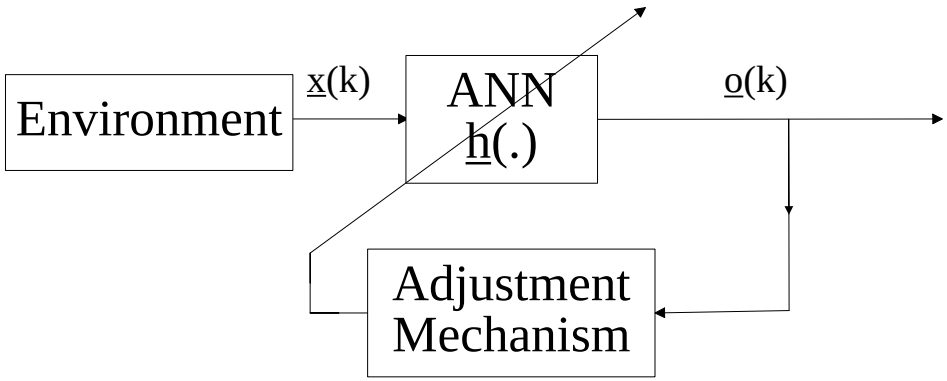


Issues

1. Suffer from speed-accuracy tradeoff, i.e. high speed of convergence $\Rightarrow$ less accuracy in parameter estimation.
2. Incremental learning requires old data.
3. Overtraining leads to performance degradation.

2. Unsupervised Learning

- Uses unlabeled data i.e training samples have no desired outcomes.
- No external teacher i.e. based upon *self-organization*.
- Form clusters to discover interesting features from data.
- Clusters are formed based upon underlying statistical properties of the data (unconditional density estimator vs conditional for supervised).
- Simpler algorithms than those of supervised.
- Used for data clustering, compression, and dimensionality reduction.

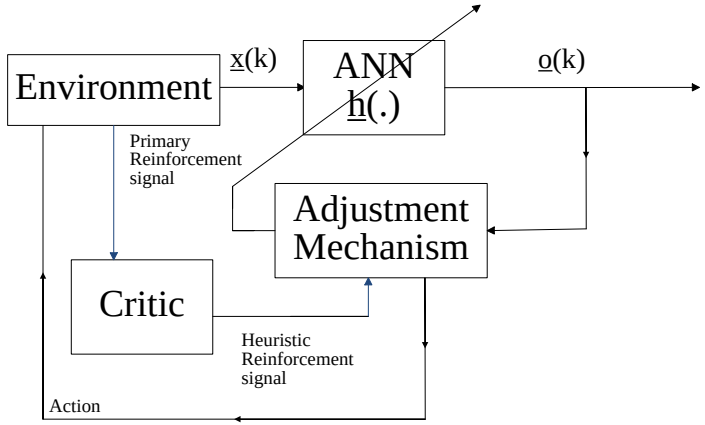


Issues

1. Requires more parameter tuning.
2. No automatic shut-off system.
3. Less accurate than supervised counterparts when used for decision-making.

3. Reinforcement Learning

- Originated in psychology with experiments on animal learning.
- No unambiguous outcome as in supervised learning i.e. no explicit supervision.
- Given training set $\{\underline{x}(k), r(k)\}_{k=1}^{K}$, where $r(k) = \{-1, +1\}$ is supplied by a "critic" (not desired signal) as a reinforcement (or appropriateness) signal.
- *Reward and Punishment-based* actions designed to maximize the expected value of a criterion function.
- Learning should devise a sequence of actions over time to obtain large *Cumulative Reward*.
- Applied in web-indexing, cell-phone network routing, control theory (e.g., robotics, unmanned vehicles), marketing strategy selection, etc.



- In machine learning, environment is typically formulated as a Markov decision process (MDP) $\Longrightarrow$ dynamic programming techniques.
- It provides feedback to the environment in forms of actions (i.e. allows for interaction).

Generalized Learning Rule (Amari 1990)

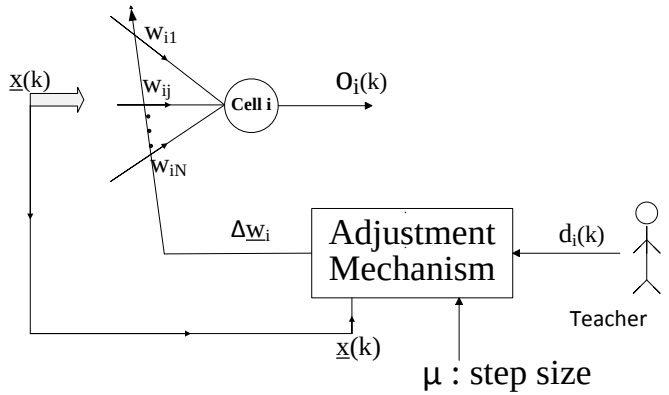
Weight vector $\underline{w}_i$ is updated in proportion to the input $\underline{x}(k)$ and a *learning* signal $r(\underline{w}_i(k), \underline{x}(k), d_i(k))$.

Thus, increment of $\underline{w}_i(k)$ at iteration k is $\Delta\underline{w}_i(k) = \mu \;\; r(\underline{w}_i(k), \underline{x}(k), d_i(k)) \; \underline{x}(k)$ where μ is learning rate (or step size).

Then, the weight vector at iteration $k+1$ is adjusted using

$\underline{w}_i(k+1) = \underline{w}_i(k) + \Delta\underline{w}_i(k)$

$\underline{w}_i(0)$'s are randomly initialized.



This provides a unified representation for several learning rules that we shall cover. In the next few lectures we cover,

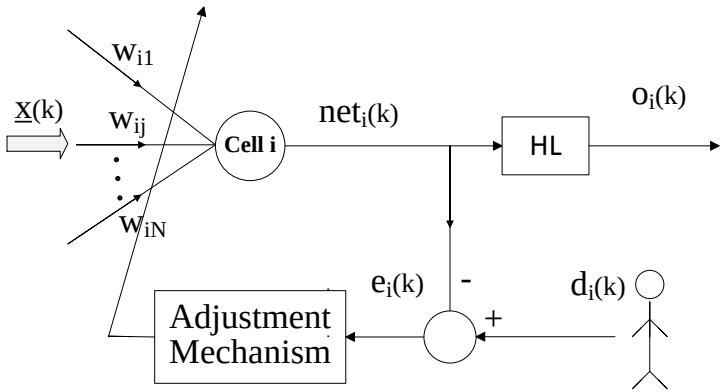
(a) **Performance Learning** - used for classification, function approximation.
(b) **Coincidence Learning** - used for association (memory learning).
(c) **Competitive Learning** - used for clustering.
(d) **Reinforcement Learning** used in controls, game theory, multi-agent systems.

and their applications in real-world problems.

a. Performance Learning-ADALINE (Widrow-Hopf 1960)

These learning methods are based upon minimizing or maximizing a performance measure (cost function).

ADALINE (ADAptive LINear Element) is based upon McCulloch-Pitts model with bipolar HL activation.



Objective: Find $\underline{w}_i$ such that MSE between net_i and the desired (supervised) is minimized.
That is, given $\{\underline{x}(k), d_i(k)\}_{k=1}^{K}$, find $\underline{w}_i^*$ to minimize,

$$J(\underline{w}_i) = \frac{1}{K}\sum_{k=1}^{K}(d_i(k) - net_i(k))^2 = \frac{1}{K}\sum_{k=1}^{K} e_i^2(k): \quad \text{Time-Averaged SE} \quad (1)$$

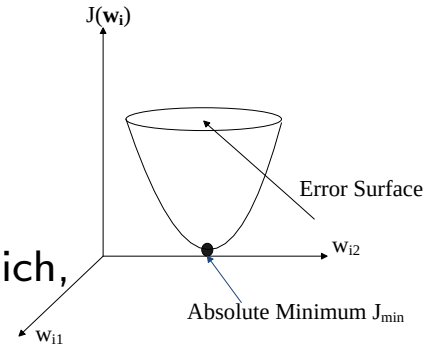
where $net_i(k) = \underline{w}_i^t \underline{x}(k)$ and $d_i(k) = net_i(k) + e_i(k)$.

For an ergodic process the time-averaged MSE can be rewritten in terms of ensemble average MSE, i.e.

$$\begin{aligned} J(\underline{w}_i) &= E[(d_i(k) - \underline{w}_i^t \underline{x}(k))^2] \\ &= E[d_i^2(k)] + \underline{w}_i^t E[\underline{x}(k)\underline{x}^t(k)]\ w_i - 2\underline{w}_i^t E[\underline{x}(k)d_i(k)] \\ &= \sigma_d^2 + \underline{w}_i^t R_{xx}\ w_i - 2\underline{w}_i^t R_{xd} \end{aligned}$$

where $E[.]$: *Expectation Operation*, $\sigma_d^2 = E[d_i^2(k)]$: variance of $d_i(k)$, $R_{xx} = E[\underline{x}(k)\underline{x}^t(k)]$: correlation matrix of data; and $R_{xd} = E[\underline{x}(k)d_i]$: cross-correlation vector between input data and desired signal.

As can be seen $J(\underline{w}_i)$ is quadratic and has a bowl-shaped surface as a function of $\underline{w}_i$ which is also referred to as *Error Performance Surface*.



This surface has a unique global minimum at which,

$$\frac{\partial J(\underline{w}_i)}{\partial \underline{w}_i} = 0 \Rightarrow 2R_{xx}\underline{w}_i - 2R_{xd} = 0$$

Solving for $\underline{w}_i^*$ gives,

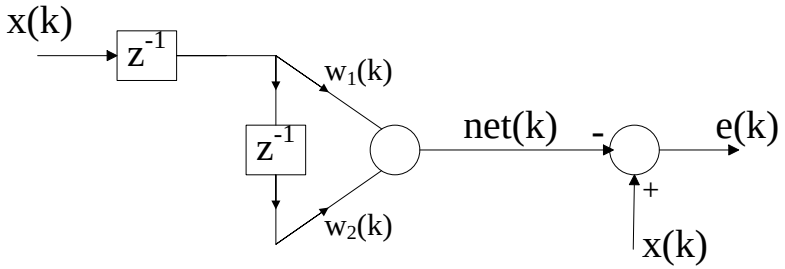
$$\underline{w}_i^* = R_{xx}^{-1} R_{xd}$$

which is the Wiener-Hopf solution (or Minimum Mean Squared Error solution (MMSE)).

Example:

Use an ADALINE to estimate weights of a 2nd order AR based linear predictor.
$x(k) = a_1 x(k-1) + a_2 x(k-2) + u(k)$, $u(k) \sim \mathcal{N}(0, \sigma_u^2)$

Here, $net(k) = \underline{w}(k)^t \underline{x}(k)$, where $\underline{x}(k) = [x(k-1)\ x(k-2)]^t$ and $\underline{w}_i(k) = [a_1(k)\ a_2(k)]^t$. Also, we choose $d(k) = x(k)$. Solving for $\underline{w}^*$ yields a_1 and a_2.



Remarks

1. Using the Wiener-Hopf solution the minimum MSE is $J_{min}(\underline{w}_i^*) = \sigma_d^2 - \underline{w}_i^{*t} R_{xd}$
2. For stationary processes the error surface has a fixed shape and orientation; whereas for non-stationary processes the bottom of the error surface moves continually and orientation and curvature may be changing too. Thus, the solution should not only seek the bottom but also track it.
3. Wiener-Hopf solution requires computing R_{xx}, R_{xx}^{-1}, and R_{xd} which makes it unsuitable for online learning.
4. For ergodic processes, solving the original time-averaged SE, which corresponds to Least Squares (LS) solution, approaches the Wiener-Hopf (i.e. MMSE) solution for large K. This is shown next.

Least Squares Solution & Geometric Interpretation

Let us reconsider the time-averaged SE in (1). Now, if we define

$\underline{d}_i = [d_i(1), \cdots, d_i(K)]^t$: Desired vector
$\underline{e}_i = [e_i(1), \cdots, e_i(K)]^t$: Error vector

$$X = \begin{bmatrix} \underline{x}^t(1) \\ \vdots \\ \underline{x}^t(K) \end{bmatrix}_{K \times N} \quad : \text{Data matrix}$$

Then we can write $\underline{d}_i = X\underline{w}_i + \underline{e}_i = \underline{net}_i + \underline{e}_i$

Now, the time-averaged SE is $J(\underline{w}_i) = \frac{1}{K}\|\underline{e}_i^2\| = \frac{1}{K}(\underline{d}_i - X\underline{w}_i)^t(\underline{d}_i - X\underline{w}_i)$
Setting $\frac{\partial J(\underline{w}_i)}{\partial \underline{w}_i} = 0$ yields
$-2X^t(\underline{d}_i - X\underline{\hat{w}}_{i_{LS}}) = 0 \implies X^tX\underline{\hat{w}}_{i_{LS}} = X^t\underline{d}_i$

which gives the LS solution,

$\underline{\hat{w}}_{i_{LS}} = (X^tX)^{-1}X^t\underline{d}_i = X^\dagger\underline{d}_i$
where $X^\dagger = (X^tX)^{-1}X^t$ is the pseudo inverse of X.

Recall from LS solution $\underline{\hat{w}}_{i_{LS}} = (X^t X)^{-1} X^t \underline{d}_i = \tilde{R}_{xx}^{-1} \tilde{R}_{xd}$ where

$$X^t X = [\underline{x}(1) \cdots \underline{x}(K)] \begin{bmatrix} \underline{x}^t(1) \\ \vdots \\ \underline{x}^t(K) \end{bmatrix} = \sum_{k=1}^{K} \underline{x}(k)\underline{x}(k)^t = \tilde{R}_{xx}\text{: Sample Correlation}$$

Matrix

$$X^t \underline{d}_i = [\underline{x}(1) \cdots \underline{x}(K)] \begin{bmatrix} d_i(1) \\ \vdots \\ d_i(K) \end{bmatrix} = \sum_{k=1}^{K} \underline{x}(k) d_i(k) = \tilde{R}_{xd}\text{: Sample}$$

Cross-correlation Vector
Clearly,

$\tilde{R}_{xx}, \tilde{R}_{xd}$: Time-averaged
R_{xx}, R_{xd} : Ensemble averaged

If data is stationary and ergodic, the LS solution asymptotically approaches the Wiener-Hopf solution for large K, i.e.

$lim_{K\to\infty} \tilde{R}_{xx} \to R_{xx}$
$lim_{K\to\infty} \tilde{R}_{xd} \to R_{xd}$

then $\underline{\hat{w}}_{i_{LS}} \to \underline{w}^*_{i_{MMSE}}$.

Now, define *Orthogonal Projection Matrix*, P_X,

$$P_X = X(X^tX)^{-1}X^t$$

which is idempotent i.e. $P_X^2 = P_X$ or $P_X^n = P_X, \forall n$ and Symmetrical i.e. $P_x^t = P_X$.

Then,

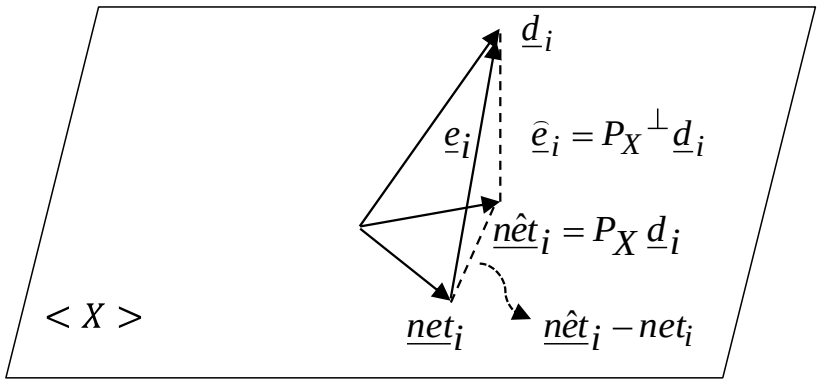
$$\underline{\hat{net}}_i = X\underline{\hat{w}}_{i_{LS}} = P_X\underline{d}_i$$

i.e. projection of $\underline{d}_i$ onto a space spanned by columns of X. Using this result the error vector at the LS solution is,

$$\underline{\hat{e}}_i = \underline{d}_i - \underline{\hat{net}}_i = \underline{d}_i - P_X\underline{d}_i = P_X^{\perp}\underline{d}_i$$

where $P_X^{\perp} = I - P_X$ is the orthogonal complement of P_X.

Note that $\underline{e}_i = \underline{\hat{e}}_i + (\underline{\hat{net}}_i - \underline{net}_i)$. The geometrical representation of these is shown in the figure.



Homework 1-Due September 15th, 2015

Problems:

1. Using the Wiener-Hopf solution show that the estimation error $e_o(k) = d(k) - \underline{w}^{*t}\underline{x}(k)$ is orthogonal to the data, i.e. $E[\underline{x}(k)e_o(k)] = E[\underline{x}(k)(d(k) - \underline{w}^{*t}\underline{x}(k))] = 0$. Explain the importance of this property.
2. Is the same orthogonality property valid for the LS solution? If it does, drive it. Compare the results in Problems 1 and 2 and their interpretations.
3. To guarantee uniqueness and improve stability of LS solution, typically a regularization term is added to the cost function, i.e. $J(\underline{w}_i) = (\underline{d}_i - X\underline{w}_i)^t(\underline{d}_i - X\underline{w}_i) + \lambda||\underline{w}_i||^2$. Find the solution of this regularized (or constrained) LS and investigate its relation to the LS solution.
4. Solve Problem 2.3 in your textbook.

b. Performance Learning-Steepest Descent

An alternative procedure using steepest (or gradient) descent method can be devised to find the weights in ADALINE without requiring to find R_{xx}^{-1}. This method involves following steps.

1. Start with an initial guess for $\underline{w}_i(0)$.
2. Use this guess to compute the gradient vector $\nabla J(\underline{w}_i(k)) \triangleq \frac{\partial J(\underline{w}_i(k))}{\partial \underline{w}_i(k)}$ at iteration (training sample) k.
3. Update weight in a direction opposite to the gradient $\underline{w}_i(k+1) = \underline{w}_i(k) - \frac{1}{2}\mu \nabla J(\underline{w}_i(k))$ where μ: Step Size
4. Go back to step (2) and repeat until convergence conditions met.

From the previous result $\nabla J(\underline{w}_i(k)) = -2R_{xd} + 2R_{xx}\underline{w}_i(k)$

Thus, $\underline{w}_i(k+1) = \underline{w}_i(k) + \mu[R_{xd} - R_{xx}\underline{w}_i(k)]$

i.e. no matrix inversion is required, but still needs to compute R_{xx} and R_{xd}.

Step size μ controls the size of incremental corrections. For convergence,

$0 < \mu < \frac{2}{\lambda_{max}}$, where λ_{max}: largest eigenvalue of R_{xx}.