USC Viterbi
School of Engineering

# MMSE ESTIMATION

## EE 541 – UNIT 3A

DR. BRANDON FRANZKE

Spring 2023

# ESTIMATION, REGRESSION, CLASSIFICATION

## statistical models

MMSE Estimation

Linear/Affine MMSE Est.

FIR Wiener filtering

Bayesian decision theory

Hard decisions

soft decisions (APP)

ML/MAP parameter estimation

Karhunen-Loeve expansion

sufficient statistics

## data driven

general regression

linear LS regression

stochastic gradient and

GD, SGD, LMS

Classification from data

linear classifier

logistical regression (perceptron)

regularization

PCA

feature design

neural networks

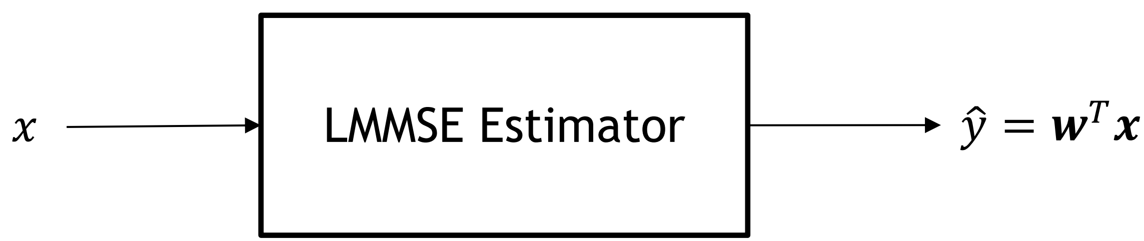
for regression and classification

learning with SGD

working with data

# PROBLEM: ESTIMATE $Y(t)$ FROM $\boldsymbol{X}(t) = \boldsymbol{x}$

- **Problem**: Given a vector observation $\boldsymbol{X}(t) = \boldsymbol{x}$, we would like to estimate $y(t)$ via linear filter $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ with minimized mean squared error (MSE).




- The objective (cost function) to be minimized is $\mathrm{MSE} = \mathbb{E}\{[y(t) - \boldsymbol{w}^T\boldsymbol{x}]^2\}$. The filter design variables are $\boldsymbol{w}$.

# ESTIMATION

- **What is Estimation?**
  - In **machine learning/signal processing/controls**, often need to make **predictions** based on **real world observations**.
  - Process known as **inference** or **estimation**.

- **What is LMMSE?**
  - **Estimates** are given as **linear combination** of observations!

# KEY IDEAS FOR RANDOM VECTORS

- Nx1 random vectors – generalization of 2x1
- Complete statistical description vs Second moment description
  - Directional preference (KL expansion)
- Gaussian processes and linear processing
- Linearity of the expectation operator

$$\mathbb{E}\{L(\boldsymbol{x}(t))\} = L(\mathbb{E}\{\boldsymbol{x}(t)\})$$

expectation commutes with any linear operation

# RANDOM VECTORS

random vector

$$\boldsymbol{X}(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_N(t) \end{bmatrix} \qquad (N \times 1)$$

**Complete** statistical description

$$p_{\boldsymbol{X}(t)}(\boldsymbol{x}) = p_{X_1(t),X_2(t),\ldots,X_N(t)}(x_1(t), x_2(t), \ldots, x_N(t))$$

*(pdf or cdf or pmf)*

$$\boldsymbol{m}_X = \mathbb{E}\{\boldsymbol{X}(t)\}$$ *mean vector*

$$\boldsymbol{R}_X = \mathbb{E}\{\boldsymbol{X}(t)\boldsymbol{X}^T(t)\}$$ *correlation matrix*

**Second Moment** Description

$$[\boldsymbol{R}_X]_{i,j} = \mathbb{E}\{X_i(t)X_j(t)\}$$

$$\boldsymbol{K}_X = \mathbb{E}\{(\boldsymbol{X}(t) - \boldsymbol{m}_X)(\boldsymbol{X}(t) - \boldsymbol{m}_X)^T\}$$ *covariance matrix*

$$= \boldsymbol{R}_X - \boldsymbol{m}_X\boldsymbol{m}_X^T$$

$$[\boldsymbol{K}_X]_{i,j} = \mathrm{Cov}[X_i(t), X_j(t)]$$

# KARHUNEN–LOÈVE (KL) EXPANSION

**Can always find orthonormal set of e-vectors of $\mathbf{K}$**

These are an alternate coordinate systems (rotations, reflections)

in this eigen-coordinate system, the components are uncorrelated

*“principal components”*

The eigen-values are the variance (energy) in each principal directions

*(reduce dimensions by "throwing out" low-energy components)*

# KL-EXPANSION

$$\boldsymbol{K}_X \boldsymbol{e}_k = \lambda_k \boldsymbol{e}_k \quad k = 1, 2, \ldots, N$$

*Eigen equation*

$$\boldsymbol{e}_k^T \boldsymbol{e}_l = \delta[k-l], \qquad \lambda_k \geq 0$$

*orthonormal eigen vectors*

$$\boldsymbol{X}(t) = \sum_{k=1}^{N} X_k(t) \boldsymbol{e}_k$$

*change of coordinates*

$$X_k(t) = \boldsymbol{e}_k^T \boldsymbol{X}(t)$$

$$\mathbb{E}\{X_k(t) X_l(t)\} = \boldsymbol{e}_k^t \boldsymbol{K}_X \boldsymbol{e}_l = \lambda_k \delta[k-l]$$

*uncorrelated components*

$$\boldsymbol{K}_X = \sum_{k=1}^{N} \lambda_k \boldsymbol{e}_k \boldsymbol{e}_k^t = \boldsymbol{E} \boldsymbol{\Lambda} \boldsymbol{E}^T$$

*Mercer's Theorem*

$$\mathbb{E}\{\|\boldsymbol{X}(t)\|^2\} = \mathrm{tr}(\boldsymbol{K}_{\boldsymbol{x}}) = \sum_{k=1}^{N} \lambda_k$$

*Total Energy*

Always exists because $\boldsymbol{K}_X$ is symmetric and positive semi- definite (PSD)

# KL-EXPANSION EXAMPLE

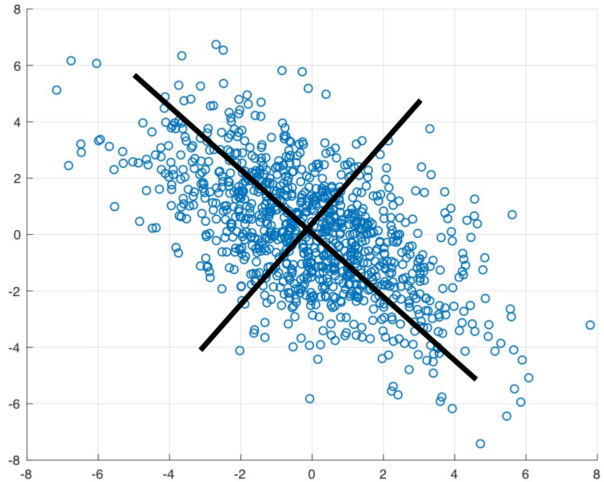
$$\boldsymbol{X}(t) = \boldsymbol{H}\,\boldsymbol{W}(t)$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix}$$

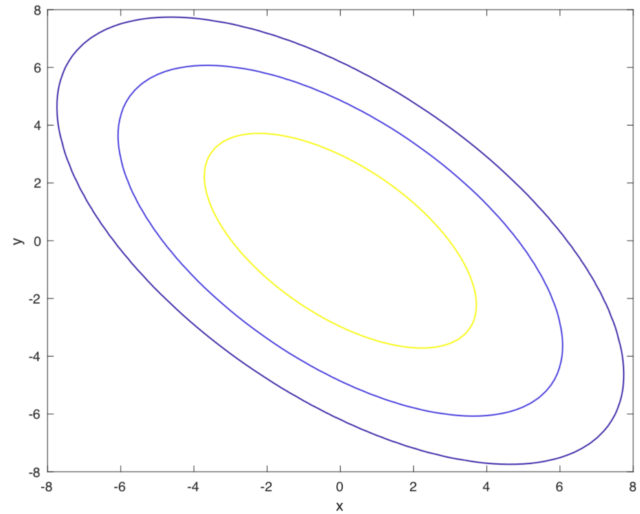
$$\boldsymbol{K} = \boldsymbol{H}\boldsymbol{K}_{\boldsymbol{W}}\boldsymbol{H}^T = \boldsymbol{H}\boldsymbol{H}^T = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

$$\boldsymbol{E} = \frac{1}{\sqrt{2}}\begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$



generated with $\boldsymbol{W}(t)$ Gaussian




Gaussian pdf contours

# LINEAR/AFFINE MMSE (LMMSE AND AMMSE)

estimate $\boldsymbol{Y}(t)$ from $\mathbf{X}(t) = x$

$$\min_{\mathrm{f(x)}} \mathrm{E}\left\{\left\|\mathrm{y(t)} - \mathrm{f}\big(\mathrm{x(t)}\big)\right\|^2\right\}$$

with no constraint, this is the conditional expectation function

$$\begin{aligned}\boldsymbol{f}_{\mathrm{opt}}(\boldsymbol{x}) &= \boldsymbol{m}_{\boldsymbol{y}|\boldsymbol{X}}(\boldsymbol{x}) \\ &= \mathbb{E}\{\boldsymbol{y}(t)|\boldsymbol{X}(t) = \boldsymbol{x}(\boldsymbol{t})\} \\ &= \int \boldsymbol{y}\, p_{\boldsymbol{y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}\end{aligned}$$

# LMMSE ASSUMPTIONS

Assume that we know the second order statistics of the joint distribution $p_{\boldsymbol{X}(t),Y(t)}(\boldsymbol{x}, y)$

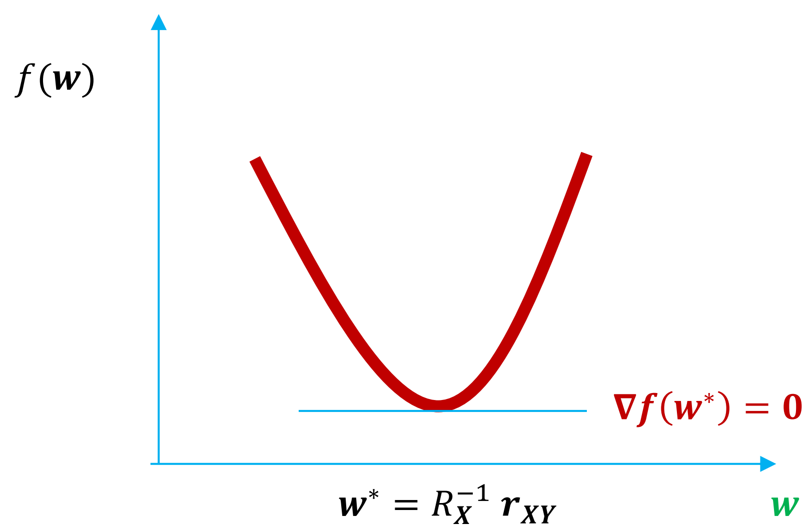
$$i.e., \boldsymbol{m}_{\boldsymbol{X}}, m_Y, R_{\boldsymbol{X}}, r_Y, \boldsymbol{r}_{XY}.$$

The objective is a quadratic function of $\boldsymbol{w}$,

$$\begin{aligned} f(\boldsymbol{w}) &= \mathbb{E}\{[Y(t) - \boldsymbol{w}^T \boldsymbol{x}]^2\} \\ &= r_Y - 2\boldsymbol{r}_{Y\boldsymbol{X}}^T \boldsymbol{w} + \boldsymbol{w}^T R_{\boldsymbol{X}} \boldsymbol{w} \end{aligned}$$

The global optimal occurs at $\boldsymbol{w}^*$.

$$i.e., \nabla f(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*} = \boldsymbol{0}$$



# MINIMUM MEAN-SQUARE ERROR ESTIMATION (MMSE)

## Estimate $\mathbf{y}(\mathbf{u})$ from $\mathbf{x}(\mathbf{u}) = \mathbf{x}$

*cross covariance matrix*

$$\mathbf{K}_{\mathbf{YX}} = \mathbb{E}\{(\mathbf{y}(t) - \mathbf{m}_{\mathbf{Y}})(\mathbf{x}(t) - \mathbf{m}_{\mathbf{X}})^{T}\} = [\mathbf{K}_{\mathbf{YX}}]^{T}$$

### Affine MMSE

$$\min_{\mathbf{f}(\mathbf{x})=\mathbf{F}\mathbf{x}+\mathbf{b}} \mathbb{E}\left\{\left\|\mathbf{y}(t) - \mathbf{f}\big(\mathbf{x}(t)\big)\right\|^{2}\right\} \qquad \mathbf{F}_{\mathrm{AMMSE}} = \mathbf{K}_{\mathbf{YX}}\mathbf{K}_{\mathbf{X}}^{-1} \qquad \mathbf{b}_{\mathrm{AMMSE}} = \mathbf{m}_{\mathbf{y}} - \mathbf{F}_{\mathrm{AMMSE}}\mathbf{m}_{\mathbf{X}}$$

$$\min_{\mathbf{F},\mathbf{b}} \mathbb{E}\{\|\mathrm{y}(t) - [\mathbf{F}\mathrm{x}(t) + \mathbf{b}]\|^{2}\} \qquad \hat{\mathbf{y}} = \mathbf{K}_{\mathbf{YX}}\mathbf{K}_{\mathbf{X}}^{-1}(\mathbf{x} - \mathbf{m}_{\mathbf{X}}) + \mathbf{m}_{\mathbf{y}} \qquad \mathrm{AMMS}\varepsilon = \mathrm{Tr}\big(\mathbf{K}_{\mathbf{Y}} - \mathbf{K}_{\mathbf{YX}}\mathbf{K}_{\mathbf{X}}^{-1}\mathbf{K}_{\mathbf{XY}}\big)$$

### Linear MMSE

$$\min_{\mathbf{f}(\mathbf{x})=\mathbf{F}\mathbf{x}} \mathbb{E}\left\{\left\|\mathbf{y}(t) - \mathbf{f}\big(\mathbf{x}(t)\big)\right\|^{2}\right\} \qquad \mathbf{F}_{\mathrm{AMMSE}} = \mathbf{R}_{\mathbf{YX}}\mathbf{R}_{\mathbf{X}}^{-1}$$

$$\min_{\mathbf{F}} \mathbb{E}\{\|\mathbf{y}(t) - \mathbf{F}\mathbf{x}(t)\|^{2}\} \qquad \hat{\mathbf{y}} = \mathbf{R}_{\mathbf{YX}}\mathbf{R}_{\mathbf{X}}^{-1}\mathbf{x} \qquad \mathrm{LMMS}\varepsilon = \mathrm{Tr}\big(\mathbf{R}_{\mathbf{Y}} - \mathbf{R}_{\mathbf{YX}}\mathbf{R}_{\mathbf{X}}^{-1}\mathbf{R}_{\mathbf{XY}}\big)$$

- affine often called linear… same when means are zero
- Conditional Expectation better than affine, affine better than Linear

# PROOF FOR LMMSE

$$\min_F \mathbb{E}\{\|y(t) - FX(t)\|^2\}$$

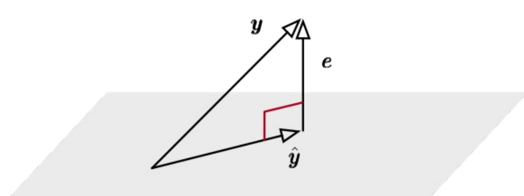
Proof:

$$\begin{aligned}
\mathrm{MSE}(F) &= \mathbb{E}\{\|y(t) - FX(t)\|^2\} \\
&= \mathbb{E}\left\{\left\|\left(y(t) - F_{opt}x(t)\right) + \left(F_{opt} - F\right)x(t)\right\|^2\right\} \\
&= \mathbb{E}\left\{\left\|y(t) - F_{opt}x(t)\right\|^2\right\} + \mathrm{Tr}\left(\left(F_{opt} - F\right)R_X\left(F_{opt} - F\right)^T\right) \\
&+ 2\,\mathrm{Tr}\left(\left(R_{yx} - F_{opt}R_X\right)\left(F_{opt} - F\right)^T\right)
\end{aligned}$$

$v^T w = \mathrm{Tr}\left(w v^T\right)$

if: $F_{opt}R_X = R_{XY}$ then: $\mathrm{MSE}(F) = \mathbb{E}\left\{\left\|y(t) - F_{opt}X(t)\right\|^2\right\} + \underbrace{\mathrm{Tr}\left(\left(F_{opt} - F\right)R_X\left(F_{opt} - F\right)^T\right)}_{\geq 0\ \forall F,\ \text{since } R_X \text{ is psd}}$

Wiener-Hopf equations, *aka Orthogonality Principle*



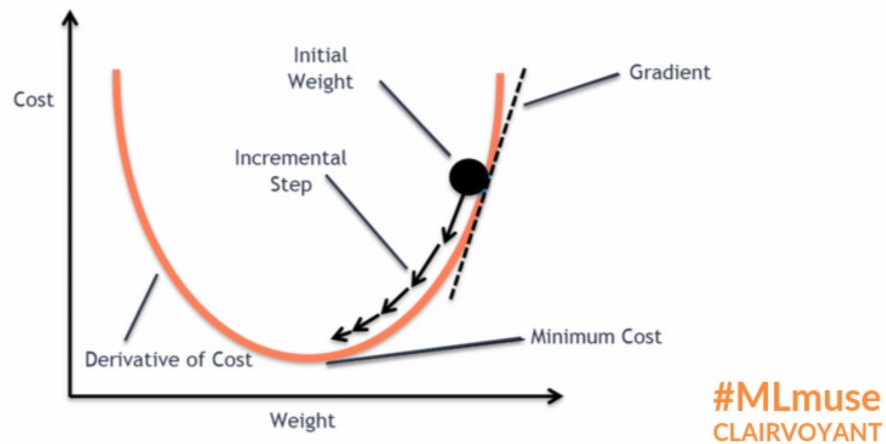

space of all estimates/approximations

because of orthogonality principle *(error and signal uncorrelated)*

$$\begin{aligned}
\mathbb{E}\{\|y(t) - \hat{y}(t)\|^2\} &= \mathbb{E}\{\|y(t)\|^2\} + \mathbb{E}\{\|\hat{y}(t)\|^2\} \\
&= \mathrm{Tr}\left(R_y - R_{yx}R_x^{-1}R_{xy}\right)
\end{aligned}$$

# GRADIENT DESCENT

- Instead of obtaining optimal $\boldsymbol{w}^*$ directly, we can also find it iteratively via
  1. Initialize $\boldsymbol{w}_0$
  2. $\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta\, \nabla f(\boldsymbol{w}_n)$

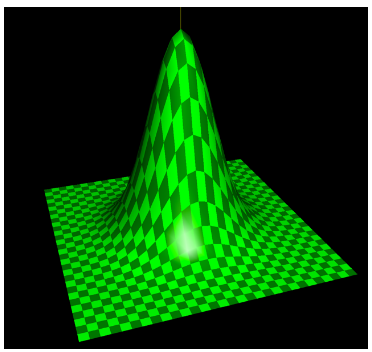



source: https://blog.clairvoyantsoft.com/

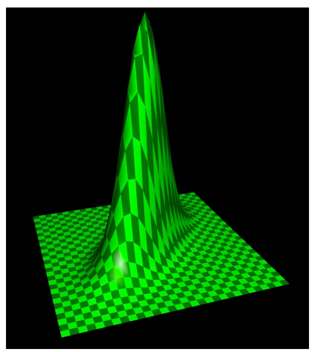
# GAUSSIAN RANDOM VECTORS

$$p_{x(t)}(x) = \mathcal{N}_N(x; m_x, K_x) = \frac{1}{(2\pi)^{N/2}\sqrt{|K_x|}} \exp\left(-\frac{1}{2}(x - m_x)K_x^{-1}(x - m_x)\right)$$

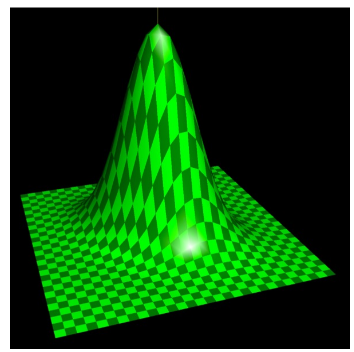
$$p_{x(t)}(x) = \mathcal{N}_N(x; 0, I) = \frac{1}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\|x - m_x\|^2\right) = \prod_{k=1}^{N} \mathcal{N}_1(x_k; 0,1)$$



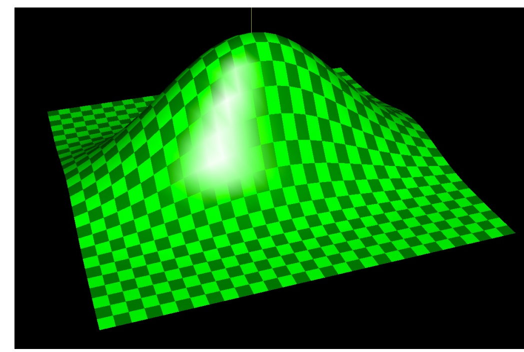
$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$K = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



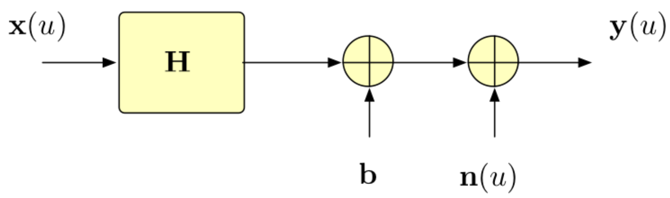
$$K = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 1 \end{bmatrix}$$



$$K = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$K = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

# GAUSSIAN RANDOM VECTORS




any linear processing of Gaussians yields Gaussians

if $\begin{bmatrix} x(t) \\ n(t) \end{bmatrix}$ is Gaussian (*i.e.*, $\boldsymbol{x}(u)$ and n$(u)$ jointly-Gaussian), then:

$y(t)$ is Gaussian

$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ is Gaussian

$\begin{bmatrix} x(t) \\ n(t) \\ y(t) \end{bmatrix}$ is Gaussian

Jointly-Gaussian common in EE!

any subset of these random variables is also Gaussian

# MMSE ESTIMATION: SPECIAL CASE JOINTLY-GAUSSIAN

## Estimate $\mathbf{y(u)}$ from $\mathbf{x(u) = x}$

Conditional Gaussian pdf:

$$p_{y(t)|x(t)}(y|x) = \frac{p_{X(t),Y(t)}(x,y)}{p_{X(t)}(x)}$$

$$= \frac{\mathcal{N}_{M+N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} m_X \\ m_Y \end{bmatrix}, \begin{bmatrix} K_Y & K_{XY} \\ K_{YX} & K_Y \end{bmatrix}\right)}{\mathcal{N}_N(x; m_X, K_X)}$$

$$= \mathcal{N}_M\left(y; \underbrace{m_Y + K_{YX}K_X^{-1}(x - m_X)}_{\text{AMMSE estimator}}, \underbrace{K_Y - K_{YX}K_X^{-1}K_{XY}}_{\text{error covariance}}\right)$$

JG… no ML or deep learning needed!

For **JG** observation and target: $\mathbb{E}[\boldsymbol{Y}|\boldsymbol{x}]$ **is the Affine MMSE estimator**

# REMARKS

1. **Closed-form global optimality** can be derived if we have a convex and differentiable cost function.

2. **Gradient descent** works in any differentiable cost function.

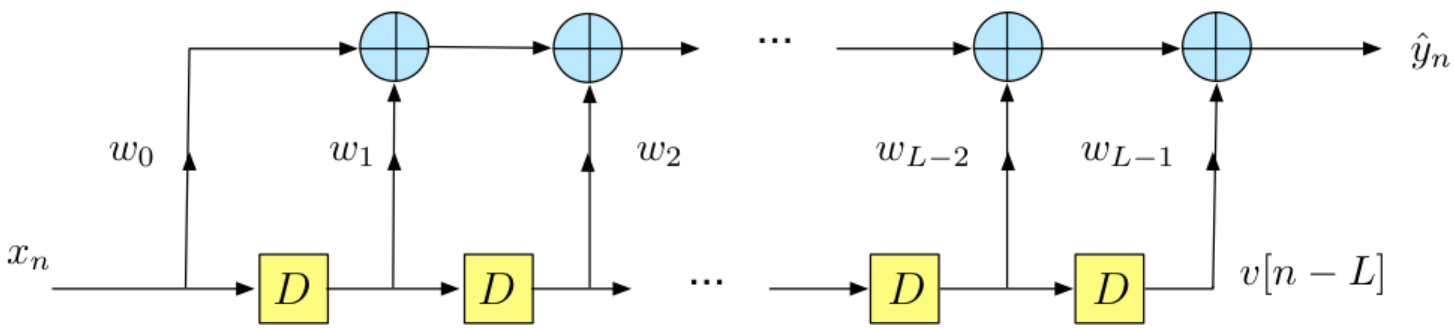
3. What if we don't have second order statistics but lots of samples? Use LMS!

   **Single point gradient descent or small batch gradient descent!**

   $R_x$ computation not tractable for large number of samples

# LMS ALGORITHM

- Suppose time index $n$ and you have samples $\{(x_n, y_n)\}$ instead of the second order statistics then we use an online learning algorithm called least mean square **adaptive filtering**
  - Introduced by Widrow and Hopf (1959).




$$y_n = \sum_{l=1}^{N} w_l x_{n-l} = w^T v_n \qquad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_L \end{bmatrix} \qquad v_n = x_{n-(L-1)} = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-L+1} \end{bmatrix}$$

# LMS ALGORITHM

- Assumptions:
  - Observe $y(n) = s(n) +$ Noise and no access to the data signal $s(n)$.
  - Have a reference noise signal $x(n)$ with strong correlation to actual noise signal
- Why we need this?:
  - Slowly drifting interfering sinusoid so notch filter is insufficient
- Benefits:
  - Behaves as an adaptive notch filter
  - The notch can be very sharp, depending upon step size

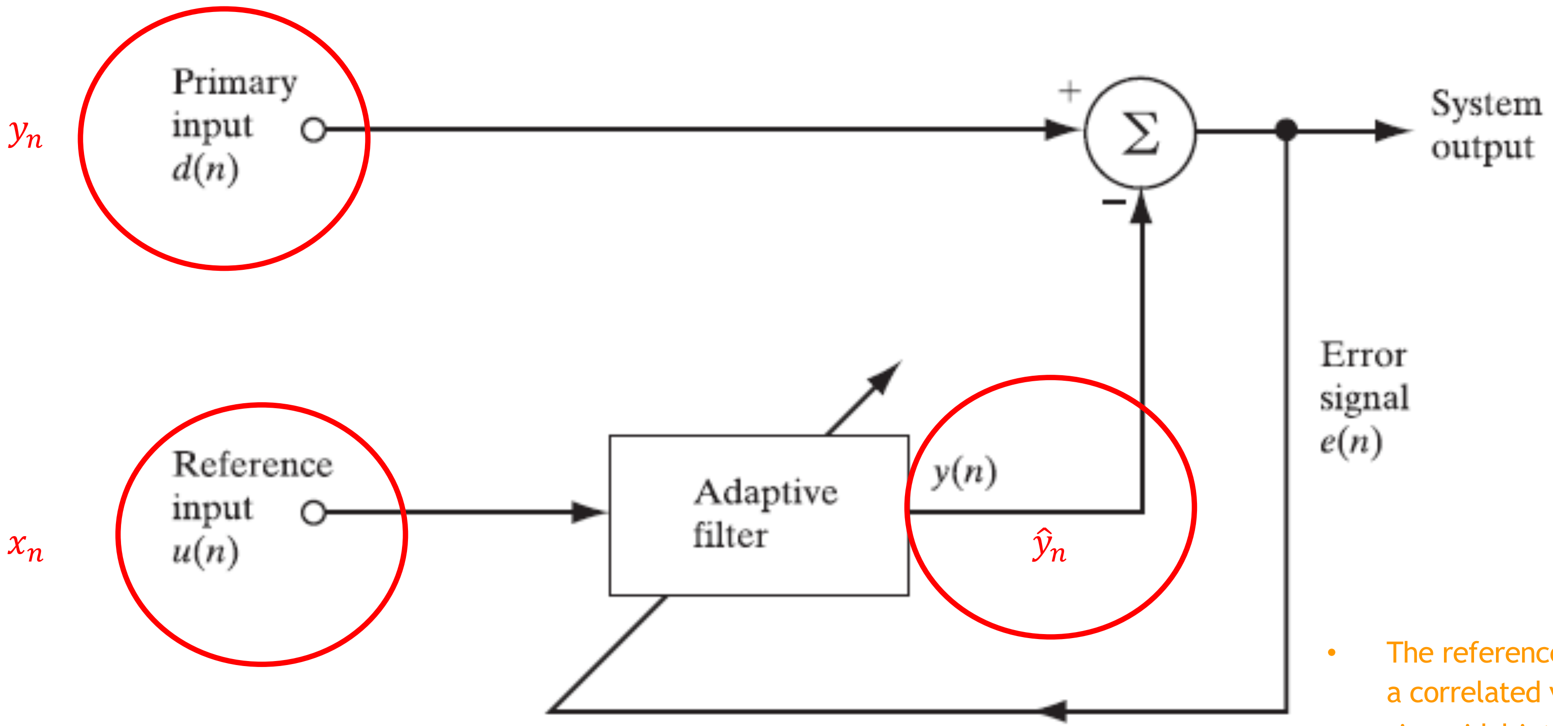


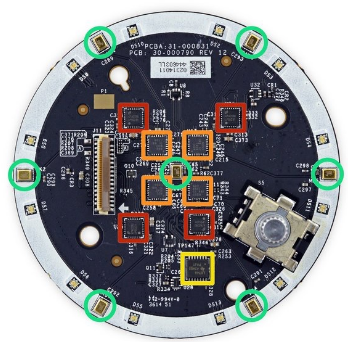

FIGURE 6.6 Block diagram of adaptive noise canceller.

- The reference input supplies a correlated version of the sinusoidal interference.
- For the adaptive filter, we may use an FIR filter whose tap weights are adapted by means of the LMS algorithm

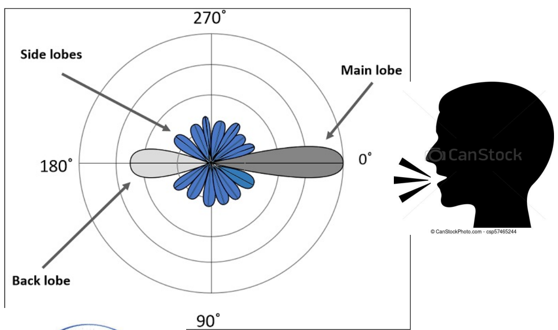
**Source**: *Simon Haykin, Adaptive Filter Theory* (5th Edition), Pearson, Section 6.3, Application 4: Adaptive Noise Cancelling Applied to a Sinusoidal Interference

# LMS HISTORY/EXAMPLE

Widrow and Hopf, Adaptive Linear Element (ADALINE)
(developed LMS for adaptive antenna array processing)



ifixit.com




point a beam at the desired speaker and ***learn***
to cancel noise energy in other directions using LMS

# STEEPEST DESCENT AND LMS

## Estimate $\mathbf{y}(\mathbf{u})$ from $\mathbf{x}(\mathbf{u}) = \mathbf{x}$

$$E(w) = \mathbb{E}\{[y(t) - w^T x(t)]^2\}$$

$$\nabla_w E = 2\mathbb{E}\{w x(t) x^T(t) - y(t) x(t)\}$$
$$= 2\left(w^t R_x - r_{xy}\right)$$

Steepest descent using (ensemble average) gradient:

$$\hat{w}_{n+1} = \hat{w}_n - \left(\frac{\eta}{2}\right) \nabla_w E$$
$$= \hat{w}_n + \eta\left(r_{xy} - R_x \hat{w}_n\right)$$

Single Point Stochastic Gradient Descent:

$$-\frac{1}{2}\nabla_w E = r_{xy} - R_X w$$
$$= \mathbb{E}\{y(t)X(t) - X(t)X^T(t)w\}$$
$$\approx y_n x_n - x_n x_n^T w$$
$$= (y_n - x_n^T \hat{w}_n) x_n$$

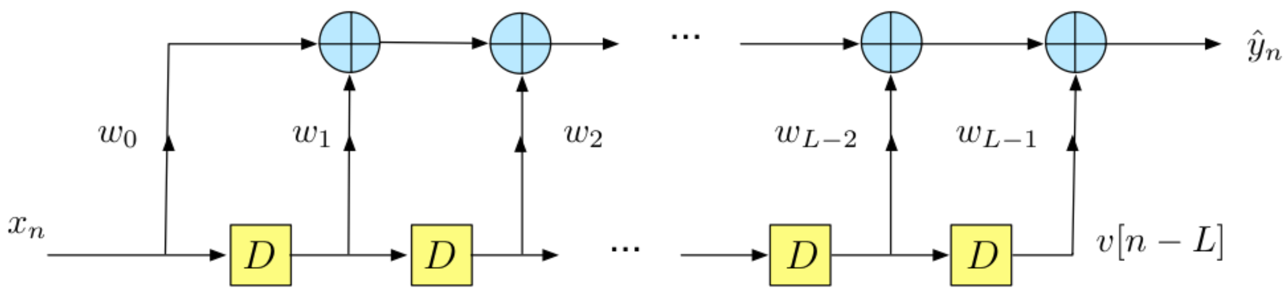
$$\boxed{\hat{w}_{n+1} = \hat{w}_n + \eta(y_n - \hat{w}_n^T x_n) x_n}$$

this is called **"on-line learning"**

when $n \sim$ time, this is the **Adaptive** Least Mean Square (LMS) filter

when $n$ does not represent time can average the gradient over more data points to improve approximation (batching)

# LMS ALGORITHM IS ADAPTIVE FIR FILTER




$$y_n = \sum_{l=1}^{N} w_l x_{n-l} \\ = w^T v_n$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

$$v_n = x_{n-(L-1)} = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-L+1} \end{bmatrix}$$

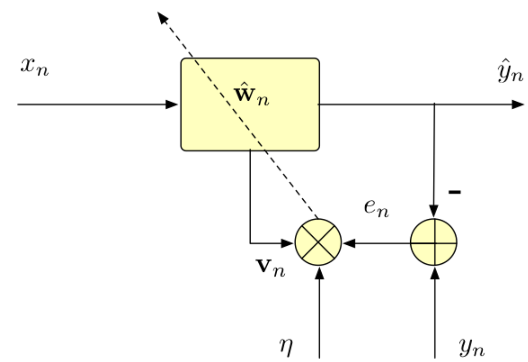
Single Point Stochastic Gradient Descent:

$$-\frac{1}{2}\nabla_w E = r_{v_n y_n} - R_{v_n} w$$
$$= \mathbb{E}\{y(t)v_n(t) - v_n(t)v_n^T(t)w\}$$
$$\approx (y_n - w_n^T v_n)v_n$$

**LMS Algorithm**

$$\boxed{\begin{aligned} \widehat{w}_{n+1} &= \widehat{w}_n + \eta(y_n - w_n^T v_n)v_n \\ &= \widehat{w}_n + \eta(y_n - \hat{y}_n)v_n \\ &= \widehat{w}_n + \eta e_n v_n \end{aligned}}$$

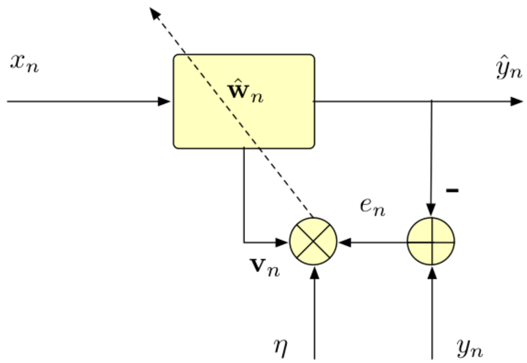
(an **online** linear regressor)



USC Viterbi
School of Engineering

# LMS ALGORITHM AS ADAPTIVE FIR FILTER

LMS algorithm: $\hat{w}_{n+1} = \hat{w}_n + \eta(y_n - w_n^T v_n)v_n$




If $R_{v_n}$ and $r_{v_n y} = \mathbb{E}\{v_n(t)y(t)\}$ do not change with $n$,

$$\hat{w}_n \rightarrow\approx w_{LMMSE} = R_v^{-1} r_{vy}$$

If these correlations vary with time, the LMS filter will **adaptively track** them

# LMS ALGORITHM: FOR SINUSOIDAL NOISE CANCELLATION

Information bearing part

Sinusoidal noise

Primary input:

$$y(n) = \boxed{s(n)} + \boxed{A_0 \cos(\omega_0 n + \phi_0)} \tag{1}$$

Reference input:

$$x(n) = \boxed{A \cos(\omega_0 n + \phi)} \tag{2}$$

Same nature as
Sinusoidal noise

$s(n)$ = **information bearing signal / data signal.**
LMS: get current estimate and error:

$$\begin{aligned} \hat{y}(n) &= \sum_{i=0}^{L-1} \hat{w}_i(n) x(n-i) \\ &= \hat{\mathbf{w}}_n^T \mathbf{v}_n \end{aligned} \tag{3}$$

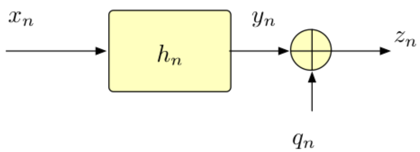
$$e(n) = y(n) - \hat{y}(n) = y(n) - \hat{\mathbf{w}}_n^T \mathbf{v}_n \tag{4}$$

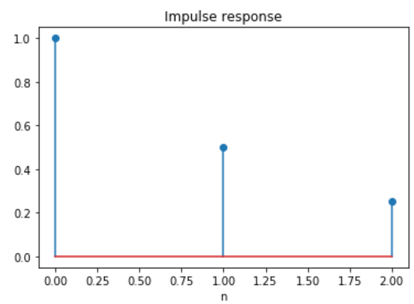
Update filter tap weights:

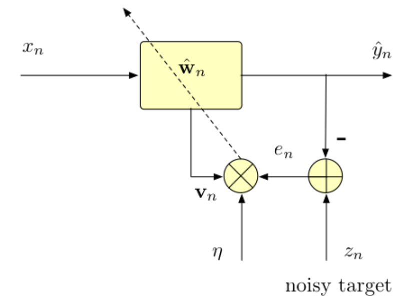
$$\begin{aligned} \hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \eta e(n) \mathbf{v}_n \\ &= \hat{\mathbf{w}}_n + \eta (y(n) - \hat{\mathbf{w}}_n^T \mathbf{v}_n) \mathbf{v}_n \end{aligned} \tag{5}$$

# EXPERIMENT: LMS EXAMPLE

Generate data:

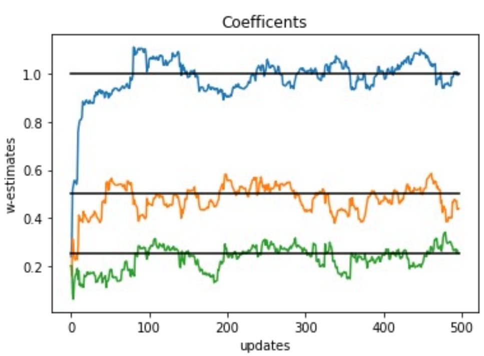


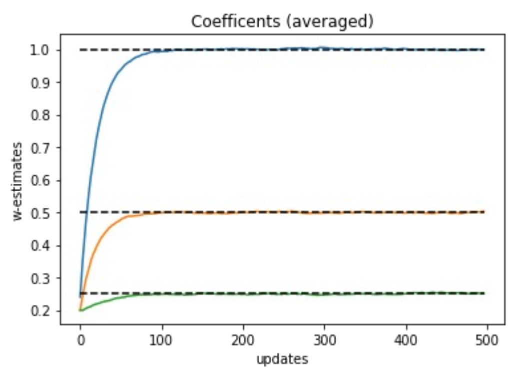




**this is the ideal case where model and observed data are matched**

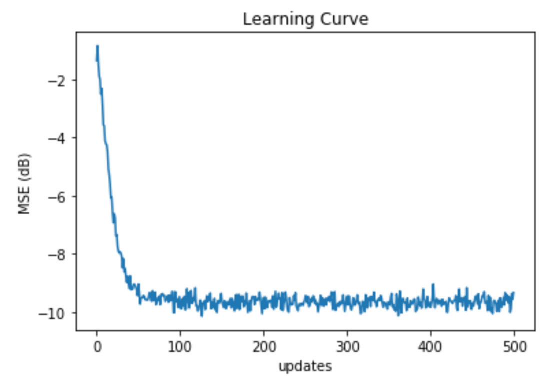
$\eta = 0.05$, SNR = 10 dB




single run



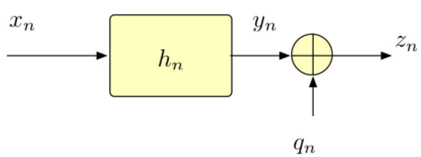

averaged over 500 runs

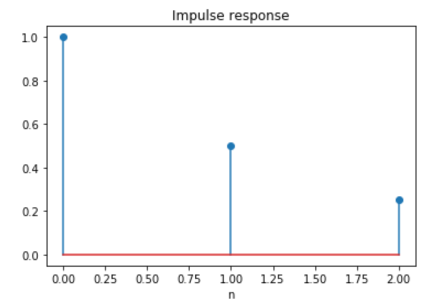


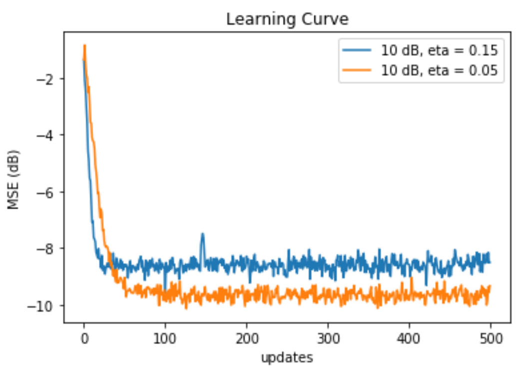

averaged over 500 runs

# EXPERIMENT: LMS EXAMPLE

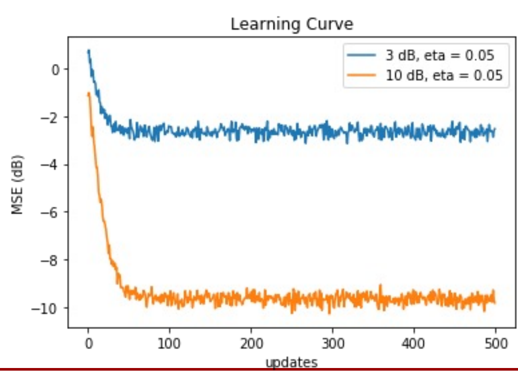
Generate data:




larger learning rate means faster convergence but more misalignment (gradient noise)




even the optimal Wiener (LMMSE) filter will have higher MMSE when the SNR is lower

# EVOLUTION OF THE LMS ALGORITHM

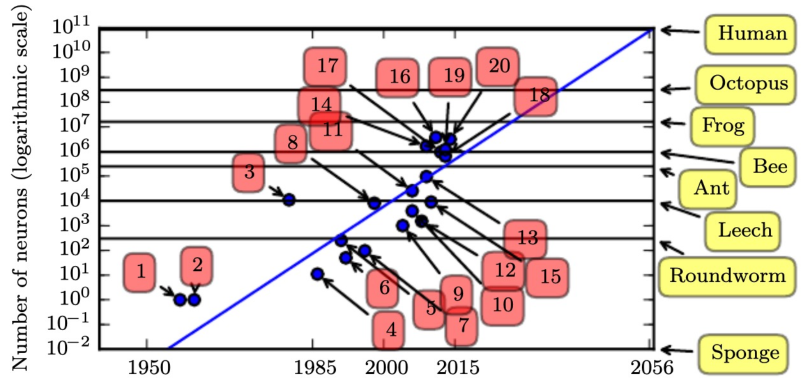



Figure 1.11: Increasing neural network size over time. Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

# MMSE SUMMARY

1. Estimation using statistical models
2. Best MMSE estimator (unconstrained) is conditional expectation
   - Requires complete statistical description of observed and desired — i.e., $p(\boldsymbol{y}|\boldsymbol{x})$
3. Linear/affine MMSE estimator have closed form equations
   - Require only the second moment description of observed and desired — i.e., means, correlations
4. For jointly Gaussian observed and desired – 2 & 3 are the same!
5. The LMS algorithm is an algorithm that approximating the LMMSE cost function gradient with a single realization.