LECTURE NOTES ON
USING NEURAL NETWORKS
FOR MEAN-SQUARED ESTIMATION.

EE 278, AUTUMN 2017
BALAJI PRABHAKAR
The Story So Far...

Consider the AWGN Channel:

\[ Y = X + Z \; ; \; X \sim \mathcal{N}(0,P), \; Z \sim \mathcal{N}(0,N), \; X \perp \!\!\!\!\!\!\!\!\!\perp Z \]

MMSE Problem: Find \( \hat{X} = g(Y) \) s.t. \( E(\hat{X} - X)^2 \) is minimized.

Solution: \( \hat{X} = g(Y) = E(X|Y) \)

The best linear estimator \( \hat{X}_{\text{lin}} = aY + b = \frac{P}{P+N}Y \), which also turns out to be the best MMSE estimator for the AWGN channel.

That is, \( E(X(Y)) = \frac{P}{P+N}Y \).
In general, estimation problems are formulated as follows:

Given \( Y = h(x,z) \), where \( x \) is signal & \( z \) is noise

find \( \hat{x} = g(y) \) s.t. \( \mathbb{E}(x-\hat{x})^2 \) is minimized. (MMSE)

We have seen that the best estimate is \( \hat{x} = \mathbb{E}(x|y) \).

However, \( \mathbb{E}(x|y) \) can be very hard to compute!

For example, consider some example \( h(\cdot, \cdot) \) & \( x, z \) distributions

(i) Linear channel: \( Y = x + z \), but \( x, z \) are not Gaussian → Already very hard!

(ii) Nonlinear channels: \( Y = \begin{cases} \frac{XZ}{X^2 + Z^2} & x, z \text{ arbitrarily distributed (including Gaussian)} \\ X^2 + Z^2 & \end{cases} \)

→ Also very hard!
(iii) **Vector Channels (Linear or Nonlinear)**

\[ y = h(x, z), \quad x, z \quad \text{arbitrarily distributed.} \]

These channels arise in many practical applications, including communications (mostly additive or multiplicative channels with Gaussian variables), biological systems (arbitrary channels & x, z distribution), e-commerce/retail data, networking, etc.
So: How can we evaluate $E(x|y)$?

Two ways to proceed:

(i) **Approximate** $E(x|y) = g(y)$ with polynomials

(ii) **Approximate** $g(.)$ using neural networks & data!

Let's look at polynomial approximations first.

Take a simple case: $y = x + E, \quad x \sim \text{Laplacian}(\frac{1}{\sqrt{p}})$, $z \sim \text{Laplacian}(\frac{1}{\sqrt{n}})$, $x \perp z$. 
ADDITIVE LAPLACIAN CHANNEL

\[ Y = X + Z \]
\[ f_X(x) = \frac{1}{2\sqrt{\rho}} \ e^{-\frac{|x|}{\sqrt{\rho}}} \]
\[ f_Z(z) = \frac{1}{2\sqrt{N}} \ e^{-\frac{|z|}{\sqrt{N}}} \]

\[ X \perp \!\!\!\!\perp Z. \]

**Note:**

\[ E(X) = 0 = E(Z) \]
\[ E(X^2) = 2\rho \ ; \ E(Z^2) = 2N \]

\[ E(X^K) = \begin{cases} 0 & \text{if } k \text{ odd} \\ 2 \cdot \frac{1}{2\sqrt{\rho}} \int_{0}^{\infty} x^K e^{-\frac{x}{\sqrt{\rho}}} \, dx = \frac{1}{\sqrt{\rho}} \cdot \frac{k!}{(\frac{1}{\sqrt{\rho}})^{k+1}} = \frac{k! \cdot \rho^{k/2}}{2} & \text{if } k \text{ even} \end{cases} \]

Similarly, \[ E(Z^K) = \begin{cases} 0 & \text{if } k \text{ odd} \\ \frac{k! \cdot N^{k/2}}{2} & \text{if } k \text{ even} \end{cases} \]
First, let's see if we can determine $E(x|y)$

$$E(x|y) = \int_{-\infty}^{\infty} x \cdot f_{x|y}(x|y) \, dx,$$  
where we use Bayes' Law.

To get $f_{x|y}(x|y)$:

$$f_{x|y}(x|y) = \frac{f_{y|x}(y|x) \cdot f_{x}(x)}{f_{y}(y)} = \frac{f(y-x) \cdot f_{x}(x)}{\int_{-\infty}^{\infty} f_{x}(u) \cdot f_{y}(y-u) \, du}.$$

Using $f_{y}(y) = \frac{1}{2\sqrt{\pi}} \cdot e^{-\frac{|y|}{2\sqrt{\pi}}}$,

$$f_{x|y}(x|y) = \frac{1}{2\sqrt{\pi}} \cdot e^{-\frac{|y-x|}{2\sqrt{\pi}}} \cdot \frac{1}{2\sqrt{\pi}} \cdot e^{-\frac{|x|}{2\sqrt{\pi}}},$$

which is highly complicated!
So let's look for polynomial approximations to $g(x)$ (where $E(x|y) = g(y)$).

1. **Linear**: $\hat{x} = ay + b$.

   Using the orthogonality principle, we get $a = \frac{1}{P_n}$, $b = \bar{y}$.

   So $\hat{x}_{\text{lin}} = \frac{P_n}{P_n} y$.

2. **Quadratic**: $\hat{x} = ay^2 + by + c$

   By the orthogonality principle, $(x - \hat{x}) \perp 1, y, y^2$

   $(x - \hat{x}) \perp 1 \Rightarrow E[(x - (ay^2 + by + c)) \cdot 1] = 0$

   $\Rightarrow \sum_0^0 - aE(y^2) - bE(y) - c = 0 \Rightarrow c = -aE(y^2)$. 


\[(x' - x) \perp y \Rightarrow E \left[ (x' - (ax^2 + by + c) \cdot y \right] = 0\]

\[\Rightarrow E(x'y) - aE(y^3) - bE(y^2) - cE(y) = 0\]

\[\Rightarrow E(x'^2 + x \cdot N) - aE(x'^2 + x^2 + 2x \cdot x') = 0 = 0\]

\[\Rightarrow E(x'^2) = b \left( E(x'^2) + E(z^2) + 0 \right)\]

\[\Rightarrow b = \frac{E(x'^2)}{E(x'^2) + E(z^2)} = \frac{2p}{2p + 2N} = \frac{p}{p + \mu}\]

\[(x' - x) \perp y^2 \Rightarrow E \left[ (x' - ax^2 - by - c) \cdot y^2 \right] = 0\]

\[\Rightarrow E(x'(x'^2 + x^2 + 2x'x) - aE(y^2) - bE(y^2) - cE(y^2) = 0\]

\[\Rightarrow E(x'^2 + x^2 + 2x'x^2) - aE(y^2) - E(y^2) = 0 - cE(y^2) = 0\]

**Substituting**

\[c = -aE(y^2) \Rightarrow 0 - aE(y^2) + a[E(y^2)]^2 = 0\]

\[\Rightarrow a(\text{E}(y^2) - \text{E}(y^2))^2 = a \cdot \text{Var}(y^2) = 0\]
Since $\text{VAR}(y^2) > 0$, (\therefore y^2 > 0 \text{ random variable and } P(y^2 > 0) \geq 1)

it must be that $a = 0$; hence $c = 0$

\[
\hat{X} = \hat{b}y = \frac{p}{p+n} y = \hat{X}_{\text{lin}}
\]

Quadratic

Cubic

Again, by orthogonality, $(X - \hat{X}) \perp 1, y, y^2, y^3$

Solving the equations (and after a lot of algebra), we get

\[
b = d = 0
\]

\[
a = \frac{\frac{2(p+n)[E(x^4) + 2pE(y^2)] - 2pE(y^6)}{2(p+n)E(y^6) - [E(y^4)]^2}}{2(p+n)}
\]

\[
c = \frac{[2p - aE(y^4)]}{2(p+n)}
\]

Where $E(x^4) = 24p^2$

$E(y^4) = 24(p^2 + pN + N^2)$

$E(y^6) = 6!(p^3 + p^2N + pN^2 + N^3)$
HIGHLY MESSY!

Now let's consider using neural networks & data.

Take $\hat{y} = h(x, z)$; $\hat{x} = g(y) = E(x|y)$

- $\rightarrow x$ $\rightarrow h(x, z)$ $\rightarrow \hat{y} = h(x, z)$ $\rightarrow$ MMSE estimator $\rightarrow \hat{x} = g(y) = E(x|y)$

Approximate this with a neural network

$\rightarrow \hat{y}$ $\rightarrow$ N.N. $\rightarrow \hat{x} \ (\hat{x} \text{ is what we want})$
WHAT IS A NEURAL NETWORK?

\[ \hat{y} \rightarrow \text{N.N.} \rightarrow \hat{x} \quad (\equiv \hat{x}) \]

\[ \begin{array}{c}
W_{11} \quad h_1 \\
W_{21} \quad h_2 \\
\vdots \\
W_{n1} \quad h_n \\
\end{array} \]

[\text{NOTE: THIS } h \text{ IS NOT DIRECTLY RELATED TO } h(x,z).]

The symbol "h" is used to denote "HIDDEN LAYER."

\[ \rightarrow \quad h_{1,\text{before}} = W^T y + b = \begin{bmatrix} W_{11} & \cdots & W_{1n} \\
W_{21} & \cdots & W_{2n} \\
\vdots & \ddots & \vdots \\
W_{n1} & \cdots & W_{nn} \end{bmatrix} \cdot \begin{bmatrix} y_1 \\
\vdots \\
y_n \end{bmatrix} + \begin{bmatrix} b_1 \\
\vdots \\
b_n \end{bmatrix} = \begin{bmatrix} \langle w_1, y \rangle + b_1 \\
\langle w_2, y \rangle + b_2 \\
\vdots \\
\langle w_n, y \rangle + b_n \end{bmatrix} \]

\[ \begin{array}{c}
W_{1} \\
W_{2} \\
\vdots \\
W_{n} \\
\end{array} \]

\[ k \times n \text{ "weight matrix"} \]

\[ R \times 1 \text{ "bias vector"} \]

\[ \begin{array}{c}
\phi \\
\phi \\
\vdots \\
\phi \\
\end{array} \]

\[ \text{rows of } W \text{ matrix.} \]
Apply ReLU (Rectified Linear Unit) type of nonlinearity.

\[ \vec{h}_{1,\text{after}} = \left[ \max \{ \langle \vec{w}_1, \vec{x} \rangle + b_1, 0 \} \right. \]
\[ \vdots \]
\[ \max \{ \langle \vec{w}_k, \vec{x} \rangle + b_k, 0 \} \left. \right] \]

\[ \vec{h}_{1,\text{after}} \text{ is input to 2nd hidden layer.} \]

And so on...

In pictures, the \( \vec{h}_{1,\text{after}} \) vector takes \( \vec{y} \) & projects it onto the affine planes defined by \((\vec{w}_i, b_i)\) & takes the projection only if \( \vec{y} \) is on the positive side of the affine hyperplane.
Thus Layer 1 classifies $\mathcal{F}$ into regions defined by the projections onto \((w_i, b_i)\) hyperplanes.

- It then passes the positive projections to Layer 2, setting negative projections to zero.

- And so on.

- Eventually, \(\mathcal{T}\) is a linear combination of the affine transformations in the network.

- We "train" the N.N. with data \((\gamma_i, \bar{x}_i)\) to get the "weights" \(w_{ij}\) and the biases \(b_{ik}\).

- Then use N.N. to estimate a new \(\mathcal{T}\) given a new \(\bar{x}\).