دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

ترکیب داده/ اطلاعات

تمرین سری اول

محمدحسین بدیعی

شماره دانشجویی 810199106

استاد : دکتر بهزاد مشیری

بهار 1399-1400

پاسخ سوال 1

❖ **پاسخ الف** – The Uncertain OWA Operator

مبنای این روش مشابه با دیگر روشهای OWA و البته با در نظر گرفتنِ اندازه‌گیری‌ها به صورت بازه‌های ریاضیاتی جایگزینِ اعداد است. این بازه‌ها در واقع نایقینی در اندازه‌گیری را نشان می‌دهند و پسوندِ Uncertain در کنار OWA در حقیقت این نایقینی را بیان می‌کند و سعی دارد با توجه به وجودِ نایقینی در مسأله، مقادیرِ بهینه‌ای برای پارامترهای مساله تعیین نماید. در ادامه این روش را با تفصیل توشیح خواهیم داد.

این اُپراتور همانطور که در درس ابزار دقیق به یاد داریم در واقع برای هر observation یا iteration یک نگاشتی از اندازه‌گیری‌ها که بُعدِ آن به تعداد سنسورهای آزمایش (n) بود را به یک اندازه‌ی یک بُعدی که حاصلِ تلفیقِ تمامیِ اندازه‌گیری‌ها بود را برای تصمیم گیری به ما خروجی می‌داد.

حال با فرض اینکه تعداد سنسورهای اندازه‌گیری برابر با n و تعداد observation یا همان iteration ها برابر با m باشد، مساله را به طور کامل بیان می‌کنیم. (طبق مقاله‌ی مطالعه شده در این پاسخنامه منظور از یک مشاهده (observation) ، یک آزمایش است.)

همانطور که گفتیم هر اندازه‌گیریِ انجام شده در یک مشاهده به صورت یک بازه مطرح می‌شود که دلالت بر نایقینی دارد. لذا اگر خروجی این اندازه‌گیری را a در نظر بگیریم آنگاه برای a داریم:

$$a = [a^L , a^U] = \{ x \mid a^L \leq x \leq a^U \}$$

از تعریفِ بالا واضح است که اگر ماکسیمم بازه‌ی بسته‌ی فوق و مینیممِ آن با هم برابر باشند آنگاه a یک عدد حقیقی خواهد شد و دیگر نایقینی در کار نخواهد بود.

حال با توجه به اینکه در مسائل owa باید خروجی‌های اندازه گیری‌ها را به ازای هر مشاهده از بزرگ به کوچک مرتب کنیم و در یک ردیف از ماتریسِ B قرار دهیم و این کار را برای دیگر مشاهدات هم انجام دهیم لذا در اینجا با بازه روبرو هستیم و بدین منظور با تعریف احتمالاتِ زیر بازه‌ها را با یکدیگر مقایسه می‌کنیم.

احتمالِ اینکه خروجیِ a (با توجه به نایقینی‌ای که با تعریفِ بازه در آن نهفته است) از b بزرگتر باشد به صورت زیر است:

$$p(a \geq b) = \max\left\{1 - \max\left(\frac{b^U - a^L}{l_a + l_b}, 0\right), 0\right\}$$

همچنین احتمالِ بزرگتر بودنِ b از a به صورت زیر تعریف می‌شود:

$$p(b \geq a) = \max\left\{1 - \max\left(\frac{a^U - b^L}{l_a + l_b}, 0\right), 0\right\}$$

لذا با توجه به تعاریفِ بالا طبیعتا در یک مشاهده، احتمالِ آنکه a_i از aj بزرگ‌تر باشد به صورتِ زیر خواهد بود:

$$p(a_i \geq a_j) = \max\left\{1 - \max\left(\frac{a_j^U - a_i^L}{l_{a_i} + l_{a_j}}, 0\right), 0\right\}, \quad j \in N$$

برای راحتیِ نگارش، $p_{ij} = p(a_i \geq a_j)$ در نظر می‌گیریم. لذا در یک مشاهده و با در نظر گرفتنِ تعداد سنسورها برابر با n، ماتریس p با ابعادِ n*n به صورتِ زیر خواهد بود:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ & & \vdots & \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$$

به طوریکه $p_{ij} \geq 0\,, p_{ij} + p_{ji} = 1\,, p_{ii} = \frac{1}{2}$ و اندیس‌های i و j نیز اعداد طبیعی هستند.

حال احتمالِ بازه‌ی a_i را از نظرِ بزرگی نسبت به دیگر خروجی‌های اندازه‌گیری را به صورتِ زیر تعریف می‌کنیم:

$$p_i = \sum_{j=1}^{n} p_{ij}, \quad i \in N$$

لذا برای مرتب نمودنِ a_i ها با توجه بزرگ بودنِ احتمالِ هر یک یعنی p_i، آنها را از بزرگ به کوچک مرتب کرده و ماتریسِ B را به ازای یک مشاهده می‌سازیم. بقیه‌ی سطرهای ماتریسِ B نیز مشابه با همین روند و بدست آوردنِ احتمالاتِ هر یک از بازه‌ها که مبینِ خروجی‌های اندازه‌گیری به ازای سنسورهای مختلف هستند را بدست آورده و تمامیِ سطرها را بدین صورت مرتب‌سازی می‌نماییم.

بازه‌ی s_k را بازه‌ی موردِ قبول یا aggregated value برای اندازه‌گیری‌هایمان در نظر گرفته که اندیسِ k در آن نشان دهنده‌ی شماره‌ی مشاهده‌ی ما است که طبق فرضی که برای تعداد مشاهدات داشتیم (تعداد مشاهدات را m آزمایش در نظر گرفتیم) لذا k نیز برای s_k یک عدد طبیعی بین 1 تا m خواهد بود. همچنین این مطلب را برای a_i ها هم تعمیم می‌دهیم و آنها را با a_{ki} بازنویسی می‌کنیم که در واقع k همان شماره‌ی مشاهده‌ی ما یا همان شماره iteration است. لذا با این تعاریف داریم:

$$a = [\, a_{ki}^L, a_{ki}^U] \quad , \quad s = [\, s_{ki}^L, s_{ki}^U] \quad , i \in N, \text{and } k = 1, 2, \ldots, m,$$

همچنین وزنها نیز به فرمتِ زیر خواهد بود:

$$v = \left\{(v_1, v_2, \ldots, v_n)^T \,|\, 0 \leq \alpha_i \leq v_i \leq \beta_i \leq 1, i \in N, \sum_{i=1}^{n} \alpha_i \leq 1, \sum_{i=1}^{n} \beta_i \geq 1\right\}$$

حال نوبت به بدست آوردنِ مقادیر وزن‌ها می‌شود. بدین منظور تعارفِ زیر را ارائه کرده و در نهایت با تعریفِ یک تابعِ هزینه، بهینه‌ترین مقدار را برای پارامترِ وزن‌ها بدست می‌آوریم. می‌دانیم برای اینکه شرط رسیدن به بازه‌ی قابلِ قبول برای خروجیِ مشاهده برقرار شود، طبیعتا می‌بایست خروجیِ اُپراتور در بازه‌ی s_k قرار گیرد. این صحبت را می‌توان به صورت فورمولِ زیر بازنویسی کرد.

$$g(a_{k1}, a_{k2}, \ldots, a_{kn}) = s_k, \quad k = 1, 2, \ldots, m$$

لذا تابع g را می‌توان به صورت یک تابع خطی از b_i ها که مرتب شده‌ی a_i طبق فورمول‌هایی قبلی بودند بازنویسی کرد. ضریب هر یک از b_i ها را وزن‌ها تشکیل خواهند داد که آن را v_i می‌نامیم. لذا خواهیم داشت:

$$\sum_{j=1}^{n} v_j\, b_{kj} = s_k, \quad k = 1, 2, \ldots, m$$

که اندیسِ k در فورمولِ بالا بیان‌گر مرحله‌ی مشاهده یا همان iterationِ مشاهدات است. با توجه به بازه‌هایی که برای b_i ها (که مرتب شده‌ی a_i ها بودند) داشتیم. می‌توان با درنظر گرفتنِ مینیم و ماکسیمم بازه به نتیجه‌ی زیر رسید.

$$\sum_{j=1}^{n} v_j b_{kj}^{L} = s_k^{L}, \quad \sum_{j=1}^{n} v_j b_{kj}^{U} = s_k^{U}, \quad k = 1, 2, \ldots, m$$

حال می‌توان خطای موجود بین خروجیِ اُپراتور و s را (که بیان‌گر بازه‌ی مطلوب است) را به صورت زیر تعریف کرد:

$$e_{1k} = \left| \sum_{j=1}^{n} v_j b_{kj}^{L} - s_k^{L} \right|, \quad k = 1, 2, \ldots, m$$

$$e_{2k} = \left| \sum_{j=1}^{n} v_j b_{kj}^{U} - s_k^{U} \right|, \quad k = 1, 2, \ldots, m$$

لذا می‌توان این مسأله را به صورت یک مسأله‌ی بهینه‌سازی و با تعریفِ یک تابعِ هزینه و تعریفِ مینیمم ارور تعریف کرد. لذا داریم:

$$\min e_{1k} = \left| \sum_{j=1}^{n} v_j b_{kj}^{L} - s_k^{L} \right|, \quad k = 1, 2, \ldots, m$$

$$\min e_{2k} = \left| \sum_{j=1}^{n} v_j b_{kj}^{U} - s_k^{U} \right|, \quad k = 1, 2, \ldots, m$$

$$s.t. \quad \alpha_j \le v_j \le \beta_j, \quad j \in N, \quad \sum_{j=1}^{n} v_j = 1$$

حال با یک حل مسأله بهینه‌سازی طبق مقاله به نتیجه‌ی زیر برای مینیمایز کردنِ وزن‌ها می‌رسیم:

$$\min J = \sum_{k=1}^{m} \left[(e_{1k}^{+} + e_{1k}^{-}) + (e_{2k}^{+} + e_{2k}^{-}) \right]$$

$$s.t. \quad \sum_{j=1}^{n} b_{kj}^{L} v_j - s_k^{L} - e_{1k}^{+} + e_{1k}^{-} = 0, \quad k = 1, 2, \ldots, m$$

$$\sum_{j=1}^{n} b_{kj}^{U} v_j - s_k^{U} - e_{2k}^{+} + e_{2k}^{-} = 0, \quad k = 1, 2, \ldots, m$$

$$\alpha_j \le v_j \le \beta_j, \quad j \in N, \quad \sum_{j=1}^{n} v_j = 1$$

$$e_{1k}^{+} \geq 0, e_{1k}^{-} \geq 0, e_{1k}^{+} \cdot e_{1k}^{-} = 0, \quad k = 1, 2, \ldots, m$$
$$e_{2k}^{+} \geq 0, e_{2k}^{-} \geq 0, e_{2k}^{+} \cdot e_{2k}^{-} = 0, \quad k = 1, 2, \ldots, m$$

مثبت و منفی بر روی هر یک از ارورهایی که پیش از این بحث کردیم، در واقع انحراف بالا (+) و پایین (-) را نسبت به مینیممِ s برای e_1 و انحراف بالا (+) و پایین (-) را نسبت به ماکسیمم s برای e_2 خواهند بود که قابل محاسبه هستند. بدین صورت با در نظر گرفتنِ معالات و مجهولات در معادله‌ی فوق و داشتِ شروطی که بر روی وزن‌ها داشتیم، هر یک از وزن‌ها را به سادگی می‌توانیم بدست آوریم.

❖ پاسخ ب – Induced OWA Operators

این نوع از اُپراتورِ owa مانند دیگر اُپراتورهای owa که قبل از این مطالعه نمودیم، برای هر observation یا iteration یک نگاشتی از اندازه‌گیری‌ها که بُعدِ آن به تعداد سنسورهای آزمایش (n) است را به یک خروجیِ یک بُعدی که حاصلِ تلفیقِ تمامیِ اندازه‌گیری‌ها می‌باشد را برای تصمیم گیری در اختیارِ ما قرار می‌دهد. در حقیقت این عملگر را توانستیم به صورت ریاضی زیر تعریف کنیم:

$$F_w(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j$$

و عرض کردیم که در صورتیکه a_i ها را از بزرگ به کوچک مرتب کنیم، آنگاه b_j برابر با j امین a_i مرتب شده است. حال عملگر القایی را بیان می‌کنیم. در این نوع اپراتور در کنارِ a_i ها که به آنها argument value می‌گوییم، یک v_i نیز تعریف می‌شود. این متغیرِ را order inducing می‌نامند. در واقع در این اُپراتور یک وکتورِ دو دویی که شامل یک argument value و یک order inducing است در اختیار داریم. مرتب‌سازی از بزرگ به کوچک بر اساس v_i ها صورت می‌گیرد و در نهایت مقدارِ b_j برابر با j امین المان مرتب شده‌ی متناظر با v_i که برابر با a_i می‌شود، خواهد بود. بدین صورت اُپراتور به شکل زیر درخواهد آمد:

$$F_W(\langle v_1, a_1 \rangle, \ldots, \langle v_n, a_n \rangle) = W^T B_v.$$

همچنین فورمولِ گسترده‌ی عبارت فوق، به صورت زیر در خواهد آمد که اندیس v-index(j) در واقع همان تناظر را که برای انتخاب a_i بر حسبِ بزرگیِ v_i بود را بیان می‌کند.

$$F_W(\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \ldots, \langle v_n, a_n \rangle) = \sum_{j=1}^{n} w_j a_{v\text{-index}(j)}.$$

در این اُپراتور حالت خاصی که بوجود می‌آید این است که فرض کنید دو جفت از المان‌های $\langle v_n, a_n \rangle$ دارای مقدارِ یکسان برای v داشته باشند آنگاه $b_j = b_{j+1}$ و برابر با میانگینِ حسابیِ a_i و a_{i+1} خواهد بود. حال این مطلب را برای این جفت المان‌ها با v یکسان تعمیم می‌دهیم و برای همه‌ی انها در پارامترِ bشان، میانگینِ حسابیِ a های متناظر با b را قرار می‌دهیم. این حالت خاصی بود که در این اُپراتور به وجود می‌آمد و می‌بایست به آن توجه می‌کردیم.

برای تعیین وزن در IOWA روشی را که مقاله مطرح نموده است همان روشِ یادگیریِ وکتورِ وزنها و تخمین‌های متوالی برای وزنها تا زمانِ همگراییِ انها در OWA است با این تفاوت که در این روش، تعیین b_j ها به همان صورت است که چندی قبل با تعریفِ v_i و a_i مطرح کردیم. در واقع در این روش یک متغیر به اسمِ لاندا تعریف کرده که وزن‌ها بر حسب این پارامتر و به صورت زیر بدست می‌آیند:

$$w_i(l) = e^{\lambda_i^{(l)}} / \sum_{j=1}^{n} e^{\lambda_j(l)}$$

و طبیعتا خروجی اپراتور در هر مرحله به صورت زیر خواهد بود که l در واقع شماره‌ی iteration تخمین را بیان می‌کند.

$$\hat{d}_k = b_{k_1} w_1(l) + b_{k_2} w_2(l) + \cdots + b_{k_n} w_n(l)$$

این روند به همراهِ آپدیت کردنِ متغیرِ لاندا همراه است و در هر مرحله این متغیر به صورتِ زیر بروزرسانی می‌شود:

$$\lambda_i(l+1) = \lambda_i(l) - B w_i(l)(b_{k_i} - \hat{d}_k)(\hat{d}_k - d_k)$$

و در هر مرحله نیز وزنها بدست آمده و نتیجتا تخمینِ بعدی پیدا می‌شود. لذا معادله‌ی فوق در هر مرحله لانداهای جدید را می‌یابد و این روند تا جایی ادامه پیدا می‌کند که لاندا همگرا شود.

$$\Delta_i = |\lambda_i(l+1) - \lambda_i(l)|$$

زمانیکه لاندا همگرا شد در واقع وزنها همگرا می‌شوند و با همگراییِ وزنها در واقع خروجیِ اُپراتور همگرا شده و جواب حاصل می‌شود. بدین صورت وزنها بدست خواهد آمد.

❖ **پاسخ پ** – Linguistic OWA Operators

در این روشِ owa بجای سرکار داشتن با مقادیرِ عددی ای که سنسورها به عنوان خروجی در اختیار ما قرار می‌دهند، با مفاهیمِ زبانی سر و کار داریم. اگرچه اعداد دقیق هستند ولی گاه در رساندنِ مفاهیم گویا نخواهند بود. لذا در این موارد می‌توان از مفاهیم زبانی یا linguistic بهره گرفت و برای هر عضوی که در مجموعه معین می‌کنیم یک تابع عضویت قرار داده (مثلا در مقاله این تابع را در یک مثال به صورت چهار عدد مشخص نموده ولی به صورت‌های مختلفی می‌نوان توابع عضویت را نشان داد) و سپس همان مفاهیمِ owa را برای این بخش مورد استفاده قرار داد.

همانطور که عرض کردیم ارزیابیِ یک متغیرِ زبانی بر اساس تابعِ عضویتِ آن خواهد بود. در این مقاله برای نشان دادنِ عضویتِ یک مفهومِ زبانی، یک چهارتایی را به صورتِ $(a_i, b_i, \alpha_i, \beta_i)$ در نظر گرفته است که دو عدد اول، بازه‌ای را نشان می‌دهند که این مفهومِ زبانی در تابعِ عضویتِ مجموعه، دارای مقدار یک است و دو عدد بعدی نیز به ترتیب از چپ به راست، عرضهای چپ و راستِ عضویت را در تابعِ این متغیرِ زبانی نشان می‌دهند. برای درکِ بیشترِ این مطلب به مثال زیر که در مقاله آورده شده است و گویای مطلبی که عرض کردیم می‌باشد توجه بفرمائید.

VH	$Very\_High$	$(1, 1, 0, 0)$
H	$High$	$(.98, .99, .05, .01)$
MH	$Moreorless\_High$	$(.78, .92, .06, .05)$
$FFMH$	$From\_Fair\_to\_Moreorless\_High$	$(.63, .80, .05, .06)$
F	$Fair$	$(.41, .58, .09, .07)$
$FFML$	$From\_Fair\_to\_Moreorless\_Low$	$(.22, .36, .05, .06)$
ML	$Moreorless\_Low$	$(.1, .18, .06, .05)$
L	Low	$(.01, .02, .01, .05)$
VL	$Very\_Low$	$(0, 0, 0, 0)$

همانطور که در بالا مشاهده می‌فرمائید، در ستونِ اول نمادِ متغیر زبانی را آورده است؛ در ستونِ دوم فرمِ گسترده‌ی متغیر را نشان داده و در ستونِ سوم عضویت را توسطِ یک چهار تایی بیان نموده که توضیحاتِ این چهارتایی را ارئه کردیم.

حال با تعریفِ چهارتایی‌های فوق کافی است که مساله‌ی خود را با تعاریفِ زیر که از این چهارتایی‌ها حاصل می‌شوند ادامه دهیم. لذا اگر $A = \{a_1, \ldots, a_m\}$ را به عنوانِ مجموعه برچسب‌های تجمیع شده در نظر داشته باشیم، آنگاه اُپراتورِ LOWA را به صورتِ زیر تعریف می‌نماییم:

$$\phi(a_1, \ldots, a_m) = W \cdot B^T = \mathcal{C}^m\{w_k, b_k, k = 1, \ldots, m\} = w_1 \odot b_1 \oplus (1 - w_1) \odot \mathcal{C}^{m-1}\{\beta_h, b_h, h = 2, \ldots, m\}$$

به صورتیکه در تعریفِ فوق، $W = [w_1, \ldots, w_m]$ بردارِ پارامترهای وزنِ مسأله را نشان می‌دهند. همچنین باید بدانیم که در این تعریف، $\Sigma_i w_i = 1$ ؛ $\beta_h = w_h/\Sigma_2^m w_k, h = 2, \ldots, m$ ؛ $B = \{b_1, \ldots, b_m\}$ می‌باشد. همچنین ماتریسِ A و B با تعریفِ مجموعه جایگشت بینِ متغیرِ برچسب‌های زبانی A با یکدیگر مرتبط می‌شوند:

$$B = \sigma(A) = \{a_{\sigma(1)}, \ldots, a_{\sigma(n)}\} \ , a_{\sigma(j)} \le a_{\sigma(i)} \ \forall \ i \le j$$

همچنین C نیز در تعریفِ مذکور، اپراتورِ convex combination می‌باشد که به ازای m=2 به صورت زیر قابلِ محاسبه خواهد بود.

$$\mathcal{C}^2\{w_i, b_i, i = 1,2\} = w_1 \odot s_j \oplus (1 - w_1) \odot s_i = s_k, \quad s_j, \ s_i \ \in \ S, \ (j \ge i)$$

k نیز در فرمول فوق طبق فورمولِ زیر محاسبه می‌شود:

$$k \ = \ min\{T, i \ + \ round(w_1 \cdot (j - i))\}$$

که اُپراتورِ round همان عملگرِ گرد گردنِ عدد است و T تعداد برچست‌های متغیر زبانی در کل مجموعه و w_1 نیز وزن اول اُپراتور که برای بزرگترین تابع عضویتِ برچسب‌ها بود می‌باشد. همچنین می‌دانیم که طبق تعاریفی که داشتیم، در فورمولِ C^2 متغیرِ s_j برابر با b_1 و متغیرِ s_i برابر با b_2 است.

❖ پاسخ ت – توضیح روش Maximum Bayesian Entropy برای بدست آوردنِ وزن‌ها

در این روش یک بردار وزنِ اولیه (prior) را به صورت زیر در نظر می‌گیریم.

$$w_1^{prior} = \beta_1, w_2^{prior} = \beta_2, \ldots, w_n^{prior} = \beta_n; \quad \sum_{i=1}^{n} \beta_i = 1$$

با تعریفِ فوق برای وزن‌های اولیه، اندازه‌ی Bayesian entropy به صورت زیر تعریف می‌شود:

$$B(W) = -\sum_{i=1}^{n} w_i \ln \frac{w_i}{\beta_i/(\beta_i)_{min}}$$

به طوریکه :

$$(\beta_i)_{min} = \min(\beta_1, \ \beta_2, \ldots, \beta_n)$$

همچنین فرض می‌کنیم که هیچ یک از بتاها صفر نیستند. لذا با توجه به این نکته می‌توان عبارت زیر را به سادگی نتیجه گرفت:

$$B(W) = -\left[\sum_{i=1}^{n} w_i \ln \frac{w_i}{\beta_i}\right] - \ln(\beta_i)_{min},$$

همچنین می‌دانیم که $\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} \beta_i = 1, w_i \geqslant 0, \beta_i > 0$ برقرار است. با توجه به Bayesian entropy measure و قانون ماکسیمم انتروپی داریم:

$$Max\ B(W) = -\left[\sum_{i=1}^{n} w_i \ln \frac{w_i}{\beta_i}\right] - \ln(\beta_i)_{min}, \quad orness(w) = \sum_{i=1}^{n} \frac{n-i}{n-1} w_i = \alpha, \qquad 0 \leqslant \alpha \leqslant 1$$

$$\sum_{i} = 1^n w_i = \sum_{i} = 1^n \beta_i = 1, \quad \beta_i, w_i \in [0, 1], \ i = (1, 2, \ldots, n)$$

لذا ما در اینجا توانستیم ماکیسمم بیزیان آنتروپی را بدست آوریم. حال به بحث محاسبه‌های وزن‌های مرحله بعد از وزن‌های اولیه می‌پردازیم.

در اینجا به یک تعریف دیگر که در واقع واگرایی وزن‌ها نسبت به W_{prior} را نشان می‌دهد می‌پردازیم:

$$D_n(W; W^{prior}) = \sum_{i=1}^{n} w_i \ln \frac{w_i}{\beta_i}$$

همچنین باید توجه داشت که در معادله‌ی فوق، عبارت $0 \ln \frac{0}{\beta_i} = 0$ را برقرار می‌دانیم.

همچنین باید به این نکته نیز توجه داشته باشیم که ماکسیمم اندازه‌ی آنتروپی بیزیان زمانی بدست می‌آید که تمامی وزن‌های بدست آمده با وزن‌های اولیه‌ای که از روی آن بدست می‌آیند با یکدیگر برابر بوده و انگاه مقدار ماکسیمم آنتروپی بیزیان برابر با $\ln(\frac{1}{\beta_i})_{min}$ خواهد بود. اثبات این بخش به سادگی در مقاله و با استفاده از تابع D که بیان‌گر واگرایی بود نشان داده شده است.

قضایای فوق و تعاریفی که ارائه دادیم برای رسیدن به وزن‌های بدست آمده از این تئوری لازم بود که البته از اثبات‌های رسیدن به آنها دوری کرده و فقط تعاریف را ارائه دادیم؛

حال پس از انجام محاسباتی که در مقاله‌های مرتبط در سایت قرار گرفته، به نتایجِ زیر برای محاسبه وزن‌ها می‌رسیم:

$$w_1[(n-1)\alpha+1-nw_1]^n = ((n-1)\alpha)^{n-1}[((n-1)\alpha-n)w_1+1]$$

تا این لحظه می‌دانیم با داشتنِ n که مبینِ تعداد سنسورها (و یا تعداد وزن‌ها) و همچنین داشتنِ orness(W) که برابر با 1-α است، می‌توانیم w_1 را محاسبه کنیم. حال به سراغ w_n می‌رویم.

$$w_n = \frac{((n-1)\alpha-n)w_1+1}{(n-1)\alpha+1-nw_1}.$$

طبق فورمولِ فوق w_n نیز از w_1 و α قابل حصول است. حال به سراغ مابقی وزن‌ها می‌رویم. مابقی وزن‌ها طبق فورمولِ زیر محاسبه خواهند شد.

$$w_j = \sqrt[n-1]{w_1^{n-j}w_n^{j-1}}$$

که $1 \leq j \leq n$ خواهد بود. البته می‌دانیم که برای j برابر با 1 یا n نتیجه‌ای حاصل نمی‌شود و لذا باید برای این دو مقدار از وزن به فورمول‌هایی که در ماقبل و پیش از محاسباتِ وزن‌های دیگر، نیاز بود محاسبه کنیم، مراجعه نماییم.

پاسخ سوال 2

❖ الف

تعداد iteration ها یا همان مشاهدات برابر با 1200 و تعداد سنسورها برابر با سه و ستونِ چهارم فایل نیز مقادیر واقعی (تخمین واقعی) را نشان می‌دهد. ابتدا فورمول‌های هر یک از خطاها را شرحِ مختصری داده و سپس نتایج را ارئه می‌کنیم.

- ✓ (MSE) Mean-squared Error

خطای MSE به صورت زیر قابل حصول است:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2$$

یعنی مقدار خروجی سنسور در هر iteration را از مقدار واقعی کم کرده و آن را به توانِ دو می‌رسانیم در نهایت این مقدار را به ازای تمامیِ iteration ها جمع نموده و بر تعداد iteration ها (n) تقسیم می‌نماییم.

خروجی‌های MSE به ترتیب برای سنسورِ اول تا سوم برابر مقادیرِ زیر از چپ به راست هستند:

```
Command Window
MSE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.078307    0.078114    0.078888

fx >>
```

شکل 1 - خروجی‌های گرفته شده از متلب برای خطای MSE

- ✓ (RMSE or RMSD) Root-mean-square Error

این خطا طبق فورمولِ زیر قابلِ حصول است که این فورمول را به تک تکِ iteration ها اعمال نموده و خروجی را رسم خواهیم نمود.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}}$$

خروجیِ گرفته شده از متلب نهایتا به صورت زیر خواهد بود.

```
Command Window
RMSE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.27983     0.27949     0.28087

fx >>
```

شکل 2 - خروجی‌های گرفته شده از متلب برای خطای RMSE

- ✓ (MAE) Mean absolute Error

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}\left|y_i - x_i\right|}{n}$$

نهایتا خروجیِ گرفته شده از متلب به صورت زیر خواهد بود

```
Command Window
MAE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.20382     0.20577     0.20933

fx >>
```

شکل 3 خروجی‌های گرفته شده از متلب برای خطای MAE

حال همه‌ی خروجی‌ها را در کنار هم رسم می‌نماییم.

```
Command Window

MSE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.078307    0.078114    0.078888

RMSE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.27983     0.27949     0.28087

MAE_Table =

  1×3 table

    sensor_1    sensor_2    sensor_3
    ________    ________    ________

    0.20382     0.20577     0.20933

fx >>
```

شکل 4 رسم 9 خروجی خطا

❖ ب – Learning OWA Operators From Observations

در این بخش که بسیار کاربردی و به نظرم در اغلبِ روش‌های مختلفِ owa کاربرد دارد، آمده‌ایم و از سه فورمولِ اساسی در مقاله‌ای که در صورتِ سوال ذکر شده است بهره‌برداری کرده و وزن‌ها را تعیین کردیم.

$$\lambda_i(l+1) = \lambda_i(l) - \beta w_i(b_{ki} - \hat{d}_k)(\hat{d}_k - d_k)$$

$$w_i = \frac{e^{\lambda_i(l)}}{\sum_{j=1}^{n} e^{\lambda_j(l)}}, \quad i = (1, n)$$

$$\hat{d}_k = b_{k1}\, w_1 + b_{k2}\, w_2 + \cdots + b_{kn} w_n$$

مبنای این روش یادگیریِ وزن‌ها است. از اثباتِ آن که در مقاله به طور کامل ذکر شده است می‌گذریم ولی به طور کلی اشاره‌ای به آن می‌نماییم. در این روش سعی شده که با یک پارامترِ واسطه با نامِ لاندا وزن‌ها را به روزرسانی کند. در این راستا در هر مرحله، واگراییِ وکتورِ تخمین را از وکتورِ واقعی با ضریبی از اختلاف وکتورِ تخمین و خروجیِ سنسور را محاسبه کرده و سپس این مقدارِ محاسبه شده را از لاندای قبلی کم می‌نماید. این عبارتی که در واقع واگرایی را به نحوی نمایش می‌دهد در یک بتا که در واقع سرعتِ به روز شدن یا به عبارتی دیگر مشتقی از اختلاف را در خود نشان می‌دهد ضرب می‌کند. در واقع با این ضریب سرعتِ بروزرسانی کنترل می‌شود. در مقاله این ضریب را در یک مثال برابر با 0.35 گرفته است و ما هم همین عدد را برای این متغیر در نظر گرفته‌ایم. در نهایت می‌دانیم الگوریتم باید پس از یک تعداد ایتریشن، مقدارِ تخمینِ خود را به مقدارِ واقعی نزدیک کند. این مختصر توضیحی از آنچه روی می‌دهد بود که توضیح دادم و می‌توانید برای روشن شدنِ بیشترِ مطلب به کامنت‌های موجود در کدِ متلب مراجعه بفرمایید. همچنین لازم به ذکر است که در این الگوریتم از یک مقدار اولیه‌ای برای لاندا استفاده می‌کنند و نقطه‌ی شروعِ آموزشِ وزن‌ها از این قسمت شروع می‌شود.

ابتدا خروجی‌های خواسته شده در سوال را از ترمینالِ متلب گرفته و نمایش داده و سپس نمودارِ تخمین و خروجی‌های واقعی به ازای تماميِ مشاهدات را ارائه می‌کنیم و در انتها یک نتیجه‌گیری کلی را خدمتتان شرح می‌دهیم.

Command Window

```
Learning_Method_from_Observations_Table =

  1×8 table
```

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Learning Method	0.4972	0.96926	0.024765	0.15737	0.12223	0.20616	0.58207	0.21177

```
fx >>
```

script Ln 38 Col 56

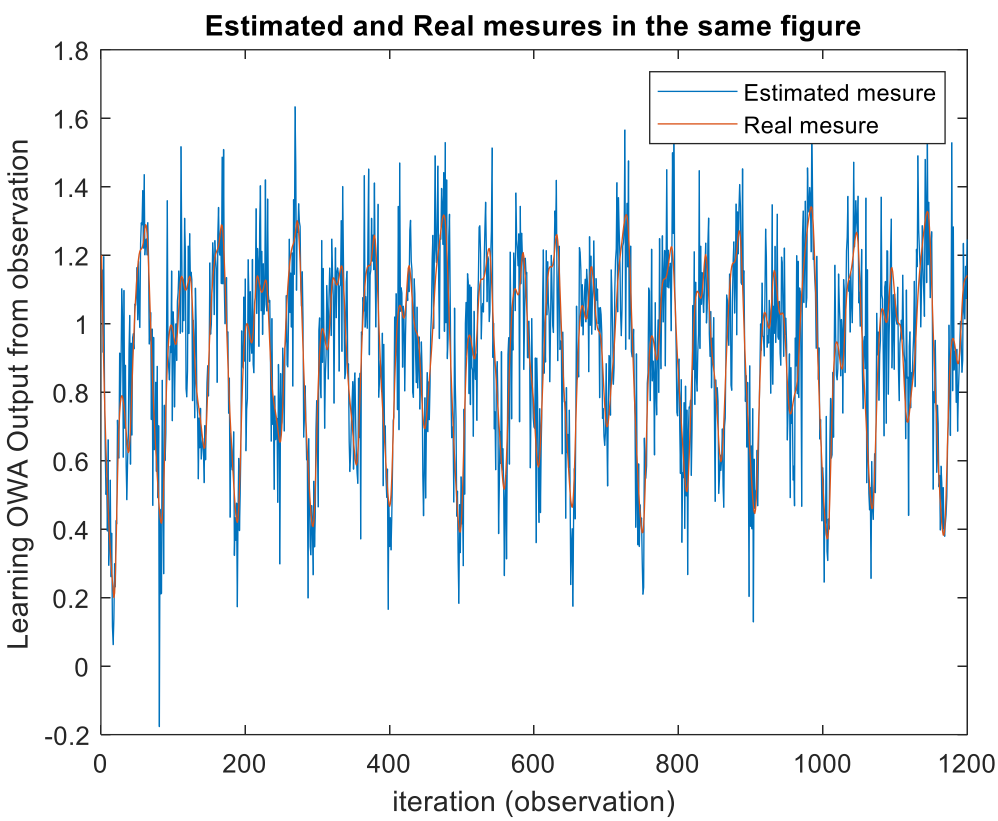
شکل 5 خروجی‌های خواسته شده در بخش Learning OWA operators from observations

جدول خواسته شده در صورت سوال را در شکلِ فوق و در ترمینال رسم نمودیم. با این حال طبق گفته‌ی سوال مجددا جدول را در زیر نگارش می‌کنیم:

Method	Orness	Dispersion	MSE	RMSE	MAE	w_1	w_2	w_3
Learning Method	**0.4972**	**0.96926**	**0.024765**	**0.15737**	**0.12223**	0.20616	0.58207	0.21177

از مقایسه جدولِ فوق با شکل 4 کاملا مشخص است که خطا تقریبا 50 درصد کاهش یافته که کاهشِ چشم گیری است و به نظرم این استدلالی بر کارایی بسیار بالای learning method می‌باشد. (در پاسخ به اینکه آیا خطا کاهش یافته است)

همچنین خواسته‌ی دیگری که سوال از ما داشته، رسم همزمانِ مقدار تخمین و مقدارِ واقعی در یک نمودار است. نمودارِ زیر خروجی گرفته شده از متلب برای پاسخ به این بخش است.



شکل 6 مقایسه‌ی اندازه‌ی تخمین و اندازه‌ی واقعی در یک نمودار در روش Learning OWA operators from observations

لذا مشاهده می‌کنیم که مقدار تخمین بسیار نزدیک به مقدار واقعی است و از طرفی در بالا نشان دادیم که ارور تا حد بسیار زیادی در هر سه روش (حدود 50 درصد) کاهش داشته است، نتیجتا با این مشاهدات می‌توان به کارایی روشِ یادگیری پی برد.

برای مشاهده پاسخ بخش پ به صفحه بعد مراجعه بفرمایید.

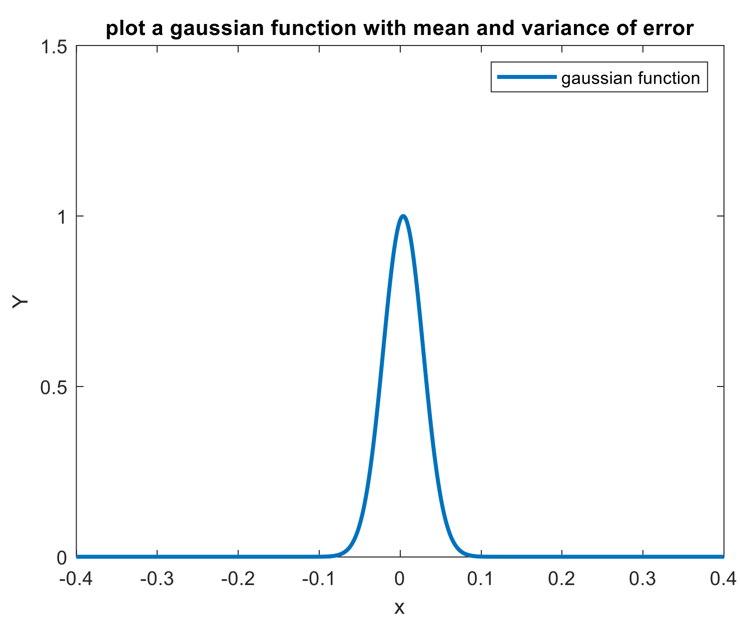
❖ بخش پ

در ابتدا سوال از ما خواسته که تابع ارور (e) را که برابر با تفاضلِ مقدارِ تخمین زده شده از مقدارِ واقعی است، تشکیل داده و سپس میانگین و واریانسِ این وکتور را محاسبه نماییم. خروجی‌ها به صورت زیر درآمدند:

```
Command Window

MEAN_VARIANCE_Table =

  1×2 table

      Mean        Variance
    _________    ________

    0.0038055     0.02475

fx >>
```

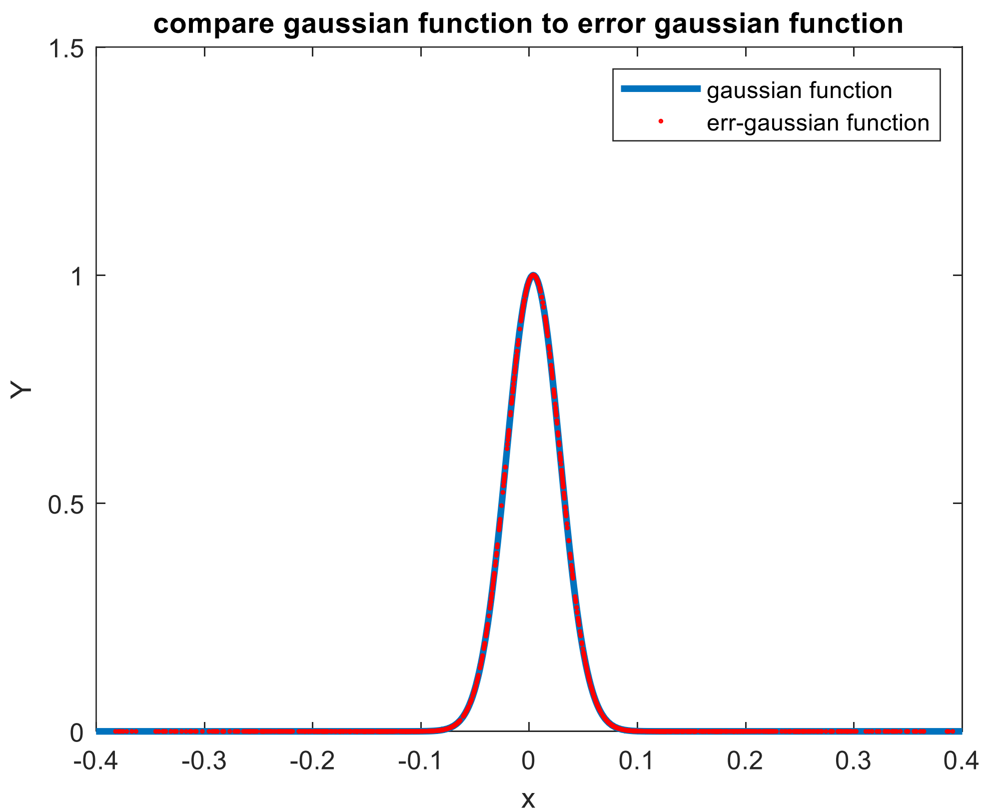
شکل 7 میانگین و واریانس محاسبه شده برای وکتورِ ارور (ارور برابر است با تفاضلِ مقدارِ تخمین از مقدارِ واقعی)

طبق خواسته‌ی مسأله تابع گوسی را با داشتنِ واریانس و میانگین رسم می‌نماییم.



شکل 8 رسم تابع گوسی با قرار دادنِ میانگین و واریانسِ وکتورِ ارور در این تابع

همچنین از ما خواسته شده است که مقادیرِ وکتورِ خطا را در تابع گوسی قرار داده و با تابع گوسی قبلی که بدست آورده بودیم در یک نمودار رسم نماییم.



شکل 9 رسم تابع گوسی با میانگین و واریانسِ وکتورِ خطا در کنار تابع گوسی که از وکتور خطا به عنوان ورودی استفاده شده

✓ تحلیل

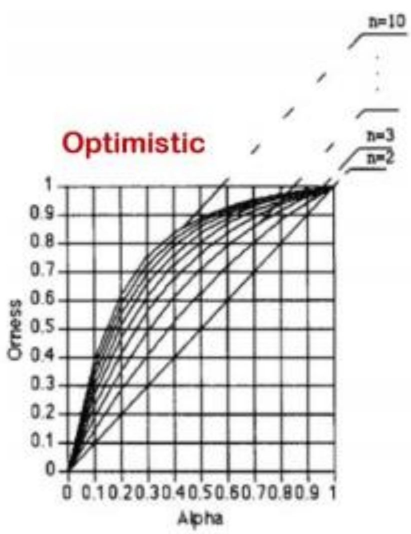
ابتدا قبل از بررسیِ شکل 9، نکات جالبی که در شکل 8 نهفته است را با هم مرور می‌کنیم. اگر مشاهده بفرمایید متوجه می‌شویم که اگرچه میانگینِ خطا نزدیک به صفر است، ولی نکته‌ی حائز اهمیت اینجاست که واریانس نسبت به میانگین حدود 6.42 (تقریبا شش برابرِ میانگین) می‌باشد. اگر این موضوع را بشکافیم به این نکته می‌رسیم که اگرچه با تعداد 1200 iteration ای که در روشِ یادگیری بکاربرده‌ایم به میانگینِ خطای بسیار کمی رسیدیم و در کل نتیجه‌ی خیلی خوبی را برای خطا به ما تحویل داده است ولی واقعیت این است که لزومِ رسیدن به این خطای کم، تعداد بالای iteration بوده و نه واریانسِ کم. یعنی ما در داده‌ها پراکندگی داریم و ممکن است اگر تعدادِ مشاهدات را پایین آوریم، دیگر نتیجه‌ای به قابلِ قبولیِ نتیجه‌ی حاضر دست نیابیم. لذا واریانسِ بالا نشان از پراکندگی داده‌ها دارد و میانگینِ خطای کم در نتایج را مدیونِ مشاهداتِ بالا هستیم. البته اگر داده‌ها واقعی باشند ممکن است این واریانسِ بالا برآمده از نویزِ موجود در سیستمِ اصلی یا حتی محیط باشد ولی چیزی که من از مشاهدات برداشت کردم این بود که outlier ای وجود ندارد (لااقل من اینطور برداشت کردم) و پراکندگی موجود که به واسطه‌ی واریانس نمایش داده شده است، احتمالا بدلیلِ وجودِ نویزِ سیستماتیک یا نویزِ محیط است نه بواسطه‌ی outlier. و اما در موردِ شکل 9 باید عرض کنم که نتیجه‌ای

که در این شکل بدست آمده کاملا قابلِ پیش‌بینی بود. در اینجا طبیعتا خروجیِ گوسین فانکسن برای نمونه‌هایی که در وکتورِ خطا موجود است، به صورتِ نقطه نقطه بر روی نمودارِ گوسینی که از قبل رسم کرده بودیم، قرار می‌گیرند و اگر غیر از این بود جای تعجب بود. ولی چیزی که هست وجودِ تراکمِ زیادِ داده‌های خطا حولِ 0.0038055 (تقریبا حول صفر) می‌باشد. یعنی نمودارِ گوسی‌ای که در شکلِ 8 بدست آورده بودیم اگر چه تنها به ورودی‌های وکتورِ خطا محدود نبود، ولی توانسته بود تخمینِ بسیار خوبی را از توزیع برای ما بزند که شکلِ 9 استدلالی بر این ادعاست.

❖ **بخش ت**

✓ روش اول – نمایی خوشبینانه (optimistic exponential)

مقدار orness را به صورت پیش فرض 0.7 در نظر می‌گیریم و پس از تعیین وزن‌ها orness مسأله را می‌یابیم.



شکل 10 نمودار orness بر حسب آلفا در روش نمایی خوشبینانه

با توجه به آنکه تعداد سنسورهای ما سه عدد است لذا مقدارِ آلفا به ازای orness=0.7 برابر 0.6 خواهد بود.

حال وزن‌ها را می‌یابیم :

✓ **Optimistic Exponential OWA Operators:**

$$w_1 = \alpha;\ w_2 = \alpha(1-\alpha);\ w_3 = \alpha(1-\alpha)^2;\ \dots;\ w_{n-1} = \alpha(1-\alpha)^{n-2};\ w_n = (1-\alpha)^{n-1}$$

W1 = 0.6   w2 = 0.24   w3 = 0.16

حال که مقادیر w(i) ها را یافتیم، می توانیم طبق دو فور مول زیر مقادیر Orness و Disp را محاسبه کنیم :

$$orness(w) = \sum_{i=1}^{n} \frac{n-i}{n-1} w_i$$

$$disp(W) = -\sum_{i=1}^{n} w_i \ln w_i$$

همچنین خروجی‌های دیگری که مثلا از ما خواسته است را در کنار مواردِ فوق در ترمینالِ متلب نمایش داده‌ایم:

Command Window

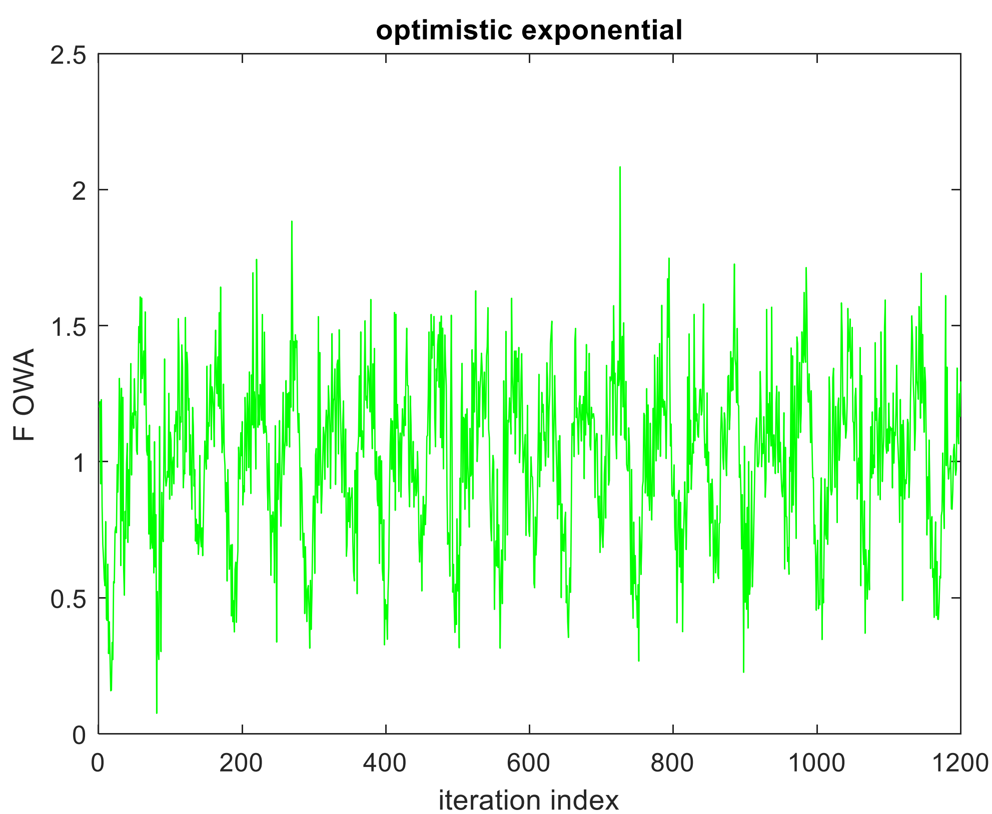
```
Optimistic_Exponential_Table =

  1×8 table
```

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Optimistic Method	0.72	0.94222	0.20383	0.45148	0.38061	0.6	0.24	0.16

```
fx >>
```

شکل 11 خروجی‌های خواسته شده‌ی سوال برای optimistic exponential

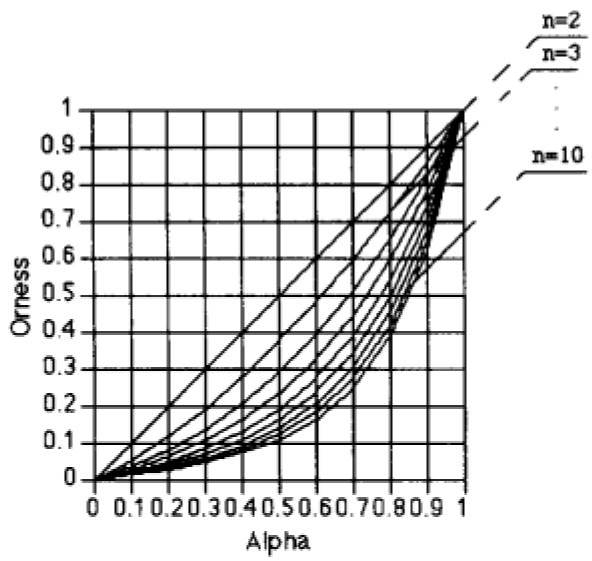
در نهایت FOWA برای بخش نمایی خوشبینانه به صورت زیر است:



شکل 12 خروجی owa برای optimistic exponential

✓ روش دوم – نمایی بدبینانه (pessimistic exponential)

مقدار orness را به صورت پیش فرض 0.7 در نظر می‌گیریم و پس از تعیین وزن‌ها orness مسأله را می‌یابیم.



شکل 13 نمودار orness بر حسب آلفا در روش نمایی بدبینانه

با توجه به آنکه تعداد سنسورهای ما سه عدد است لذا مقدارِ آلفا به ازای orness=0.7 برابر 0.77 خواهد بود.

✓ **Pessimistic Exponential OWA Operators:**

$$w_1 = \alpha^{n-1};\ w_2 = (1-\alpha)\alpha^{n-2};\ w_3 = (1-\alpha)\alpha^{n-3};\ \ldots;\ w_{n-1} = (1-\alpha)\alpha;\ w_n = (1-\alpha)$$

W1 = 0.5929 w2 = 0.1771 w3 = 0.23

خروجی‌ها در این روش به صورت زیر بدست آمدند:

Command Window

Pessimistic_Exponential_Table =

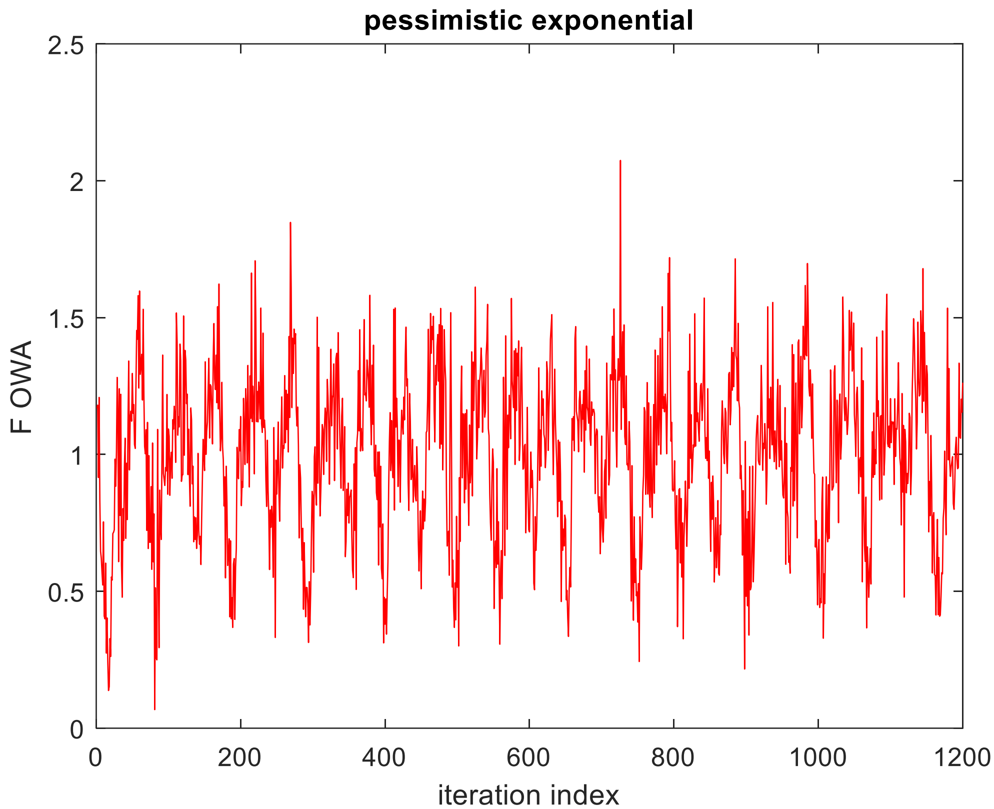
1×8 table

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Pessimistic Method	0.68145	0.95452	0.17994	0.4242	0.34978	0.5929	0.1771	0.23

fx >>

شکل 14 خروجی‌های خواسته شده‌ی سوال برای pessimistic exponential

در نهایت Fowa برای بخش نمایی بدبینانه به صورت زیر است:



شکل 15 خروجی owa برای pessimistic exponential

- ✓ روش سوم – وزن‌دار (WOWA)

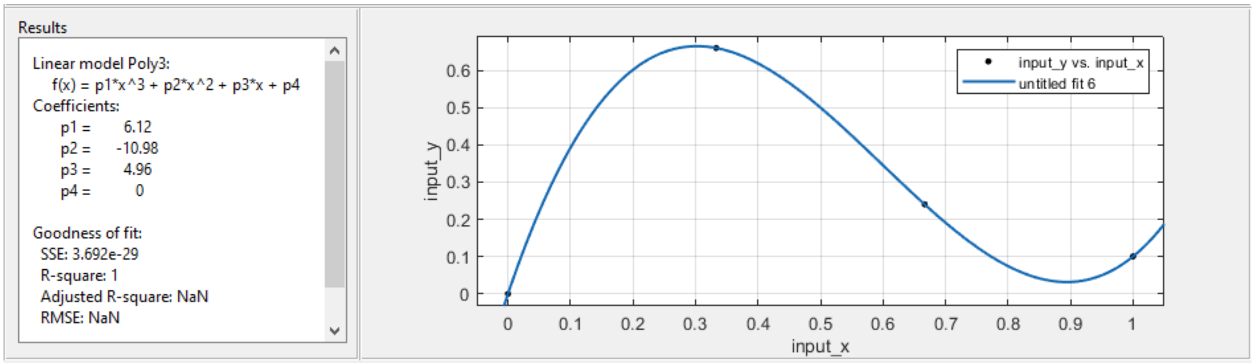
ابتدا به p هایی که در مقاله ذکر شده است مقدار دهی می کنیم (پس از چندین مقدار دهی به اعداد نهایی زیر رسیدیم و همچنین باید توجه داشته باشیم که جمع شان می بایست برابر با یک شود)

```
p = [0.2 0.1 0.7]
```

مقدار دهی اولیه وزن ها را به صورت زیر می دهیم که درجه orness آن برابر 0.78 می باشد.

```
init_w = [0.66 0.24 0.1]
```

حال یک polynomial از درجه سه به آن فیت می کنیم.



شکل 16نتایج و ضرایبِ fitting

حال w^* را طبق نتایجِ fitting پیاده سازی می کنیم. سپس طبق فورمول زیر که از مقاله استخراج شدهاست وزن ها را می یابیم :

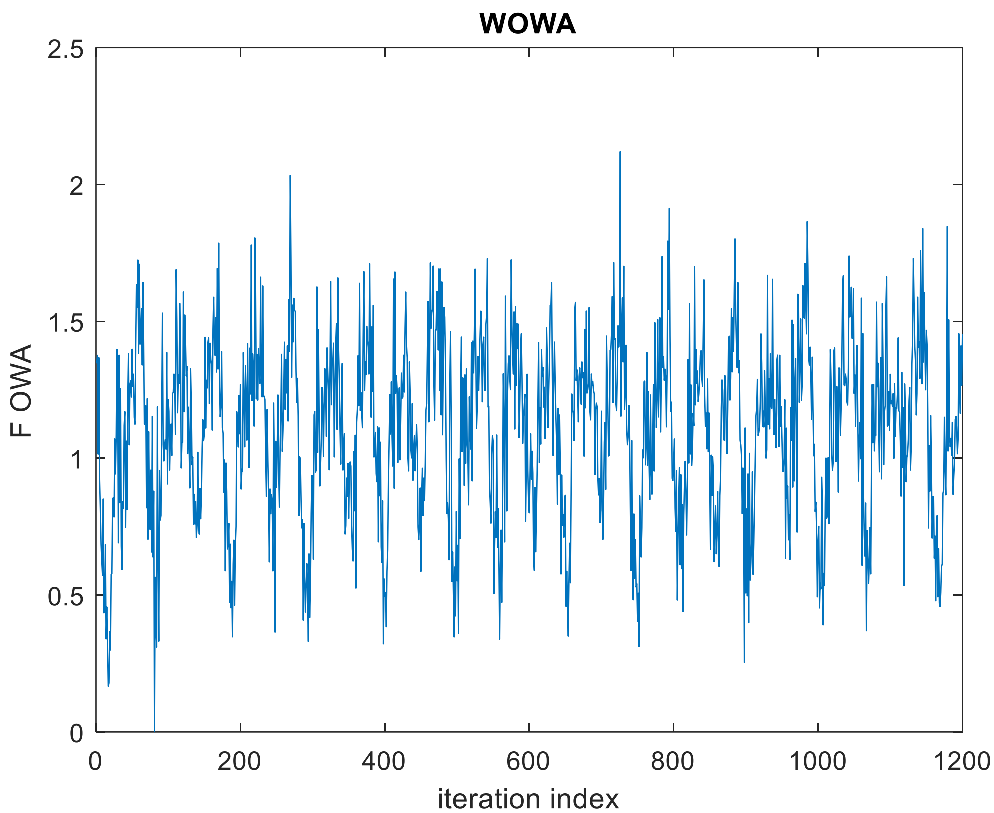
$$\omega_i = w^*\left(\sum_{j\leq i} p_{\sigma(j)}\right) - w^*\left(\sum_{j<i} p_{\sigma(j)}\right)$$

در نهایت خروجی‌های خواسته شده‌ی سوال به صورت زیر خواهند بود.

```
Command Window

WOWA_Table =

  1×8 table

                   Orness    Dispersion     MSE       RMSE       MAE        w1         w2       w3
                   _______   __________   _______   _______   _______   _______   _______   ___

    WOWA Method    0.76456    0.92149     0.34924   0.59096   0.53819   0.52752   0.47408   0.1

fx >>
```

شکل 17 خروجی‌های خواسته شده‌ی سوال برای WOWA

حال خروجیِ اپراتور به ازای مشاهدات مختلف را در یک نمودار در صفحه بعد رسم می‌کنیم.



شکل 18 خروجی owa برای WOWA

- ✓ روش چهارم – وابسته (dependent)

توابع این قسمت به طور کامل پیاده سازی شده است و باید توجه داشت که در اینجا به ازای هر iteration، w های منحصر به فرد و نتایجِ مخصوص به خودش را داریم لذا ما نتایج را برای آخرین iteration ارائه می‌دهیم. البته باید ذکر شود که چون تخمین نهایی را با تاثیرِ کل iteration ها داریم لذا خطاها منحصر بفرد و مخصوص به هر iteration نیستند؛ بلکه خطای کل مجموعه نسبت به مقادیر واقعی و مانند بقیه‌ی روش‌ها محاسبه شده‌اند. خروجی‌های خواسته شده‌ی سوال به صورت زیر هستند.

Command Window

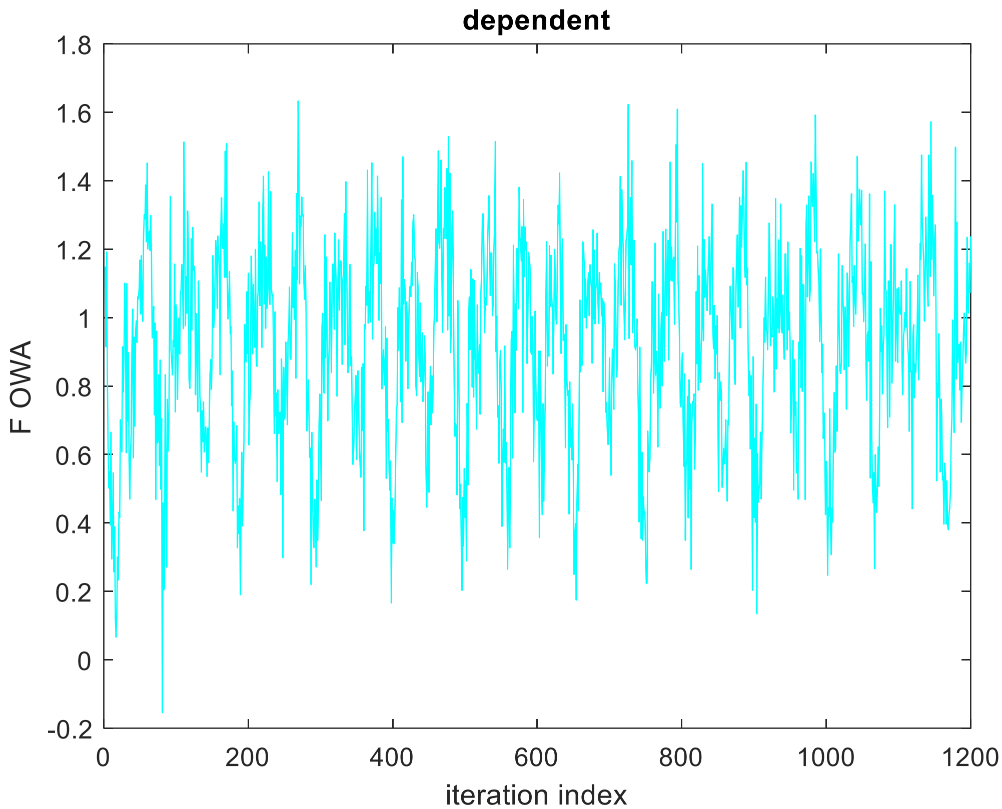
```
Dependent_Table =

  1×8 table
```

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Dependent Method	0.55046	1.0806	0.16345	0.40429	0.32778	0.35091	0.39909	0.25

```
fx >>
```

شکل 19 خروجی‌های خواسته شده‌ی سوال برای dependent

حال خروجیِ اپراتور به ازای مشاهدات مختلف را در یک نمودار رسم کردیم که به صورت زیر است:



شکل 20 خروجی owa برای روش dependent

✓ روش پنجم – روش O'Hagan

در این ورش مقدار پیش فرض را برای orness برابر 0.7 می گذاریم سپس وزنها را از جدول استخراج می‌کنیم.

با مراجعه به جداول موجود در مقاله با توجه به orness =0.7 و تعداد سنسور ها که برابر سه عدد می باشد ، سطر اول جدول زیر مناسب برای وزندهی خواهد بود.

$CF = 0.7$

# ARG.	W_1	W_2	W_3	W_4	W_5	W_6
3	0.5540	0.2921	0.1540			
4	0.4614	0.2756	0.1646	0.0984		
5	0.3962	0.2574	0.1672	0.1086	0.0706	
6	0.3475	0.2398	0.1654	0.1142	0.0788	0.0544

شکل 21 جدولِ موجود در مقاله برای تعیینِ وزن O'Hagan

پارامترهای بدست آمده به صورتِ زیر می‌باشند.

Command Window

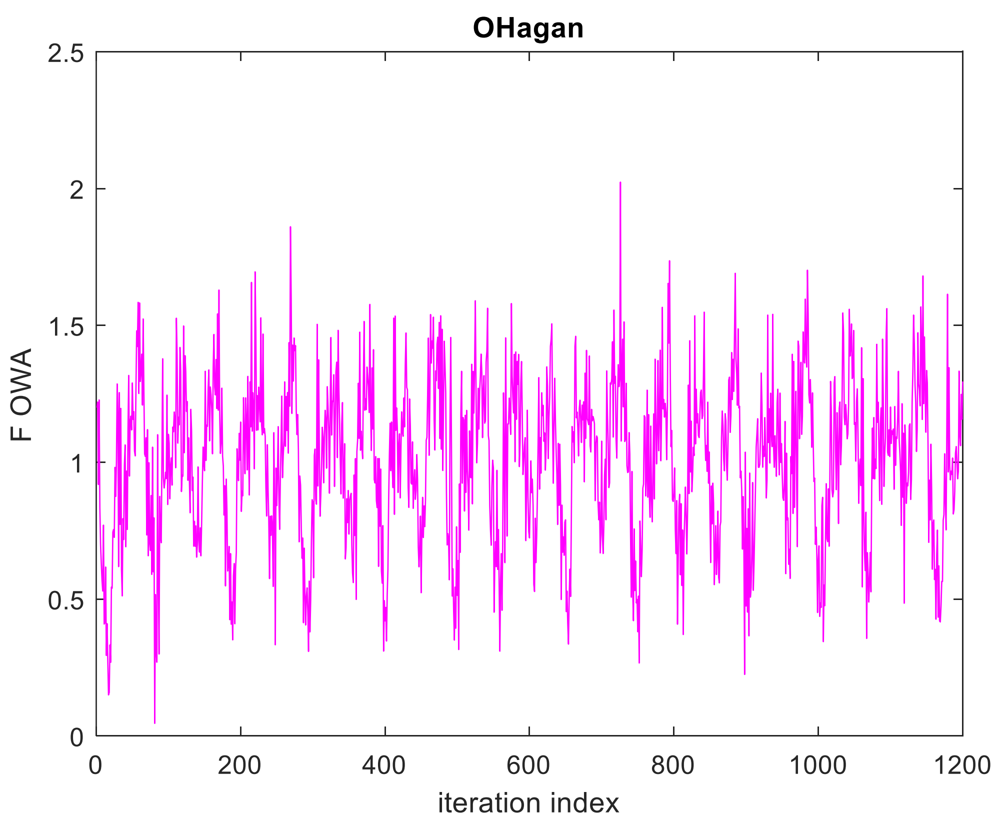
```
OHagan_Table =

  1×8 table
```

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
OHagan Method	0.70005	0.97477	0.20382	0.45146	0.3806	0.554	0.2921	0.154

```
fx >>
```

شکل 22 پارامترهای استخراج شده از روش OHagan

همچنین خروجیِ اپراتور به ازای مشاهدات مختلف نیز به صورت زیر خواهد بود.



شکل 23 خروجی owa برای روش O’Hagan

✓ روش ششم – learning

این روش را برای آپدیت کردنِ وزنها فرا گرفتیم؛ حال از این روش استفاده می‌کنیم و تخمینِ سیستم رو با توجه به این روشِ یادگیری، بدست آورده و پارامترها را تعیین می‌کنیم.

Command Window

```
Learning_Method_from_Observations_Table =

  1×8 table
```

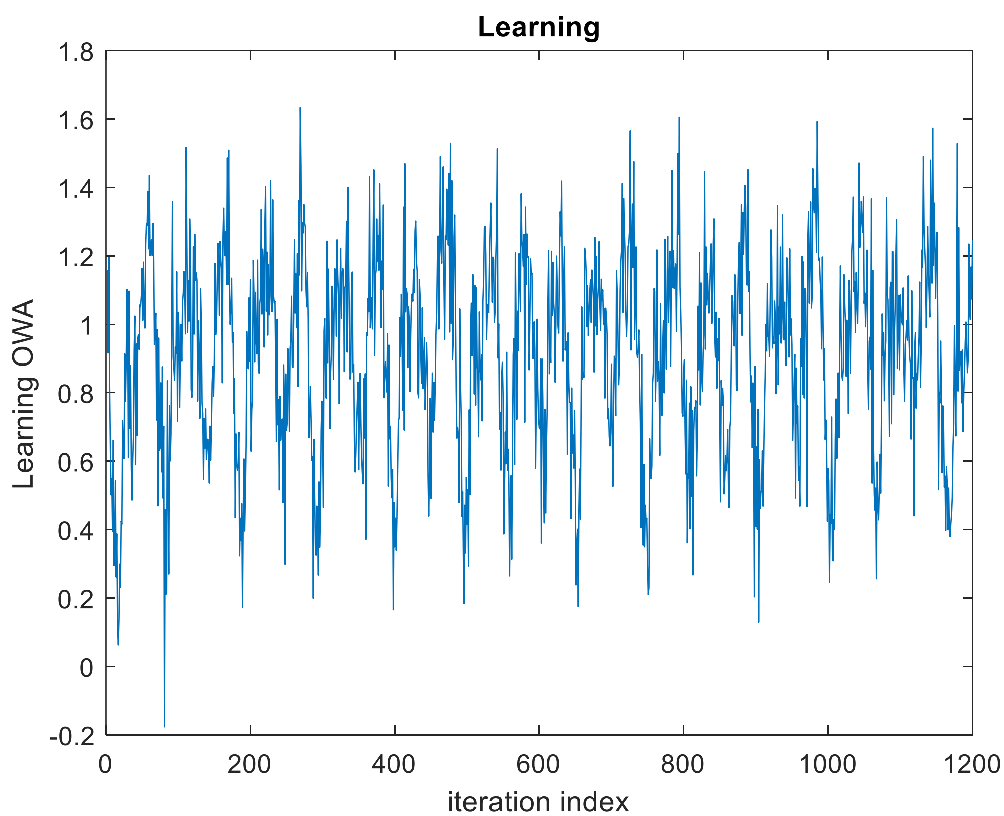
	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Learning Method	0.4972	0.96926	0.024765	0.15737	0.12223	0.20616	0.58207	0.21177

```
>>
```

script | Ln 38 Col 56

شکل 24 پارامترهای استخراج شده از روش learning

خروجیِ این اپراتور نیز به ازای مشاهدات مختلف، به صورت زیر خواهد بود.



شکل 25 خروجی owa برای روش learning

طبق خواسته‌ی سوال جدولِ مورد نظر برای هر شش روش را در متلب رسم می‌کنیم.

Command Window

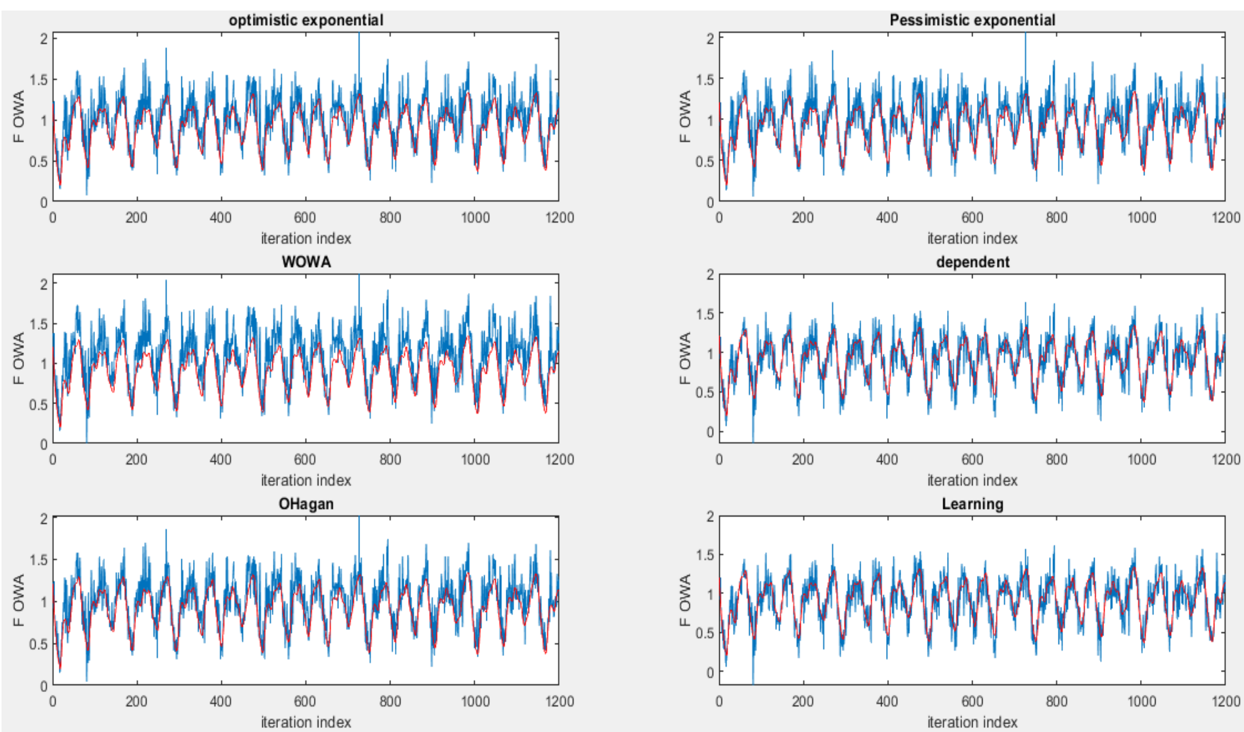
```
Methods_Table =

  6×8 table
```

	Orness	Dispersion	MSE	RMSE	MAE	w1	w2	w3
Optimistic	0.72	0.94222	0.20383	0.45148	0.38061	0.6	0.24	0.16
Pessimistic	0.68145	0.95452	0.17994	0.4242	0.34978	0.5929	0.1771	0.23
WOWA	0.76456	0.92149	0.34924	0.59096	0.53819	0.52752	0.47408	0.1
dependent	0.55046	1.0806	0.16345	0.40429	0.32778	0.35091	0.39909	0.25
OHagan	0.70005	0.97477	0.20382	0.45146	0.3806	0.554	0.2921	0.154
Learning	0.4972	0.96926	0.024765	0.15737	0.12223	0.20616	0.58207	0.21177

```
fx >>
```

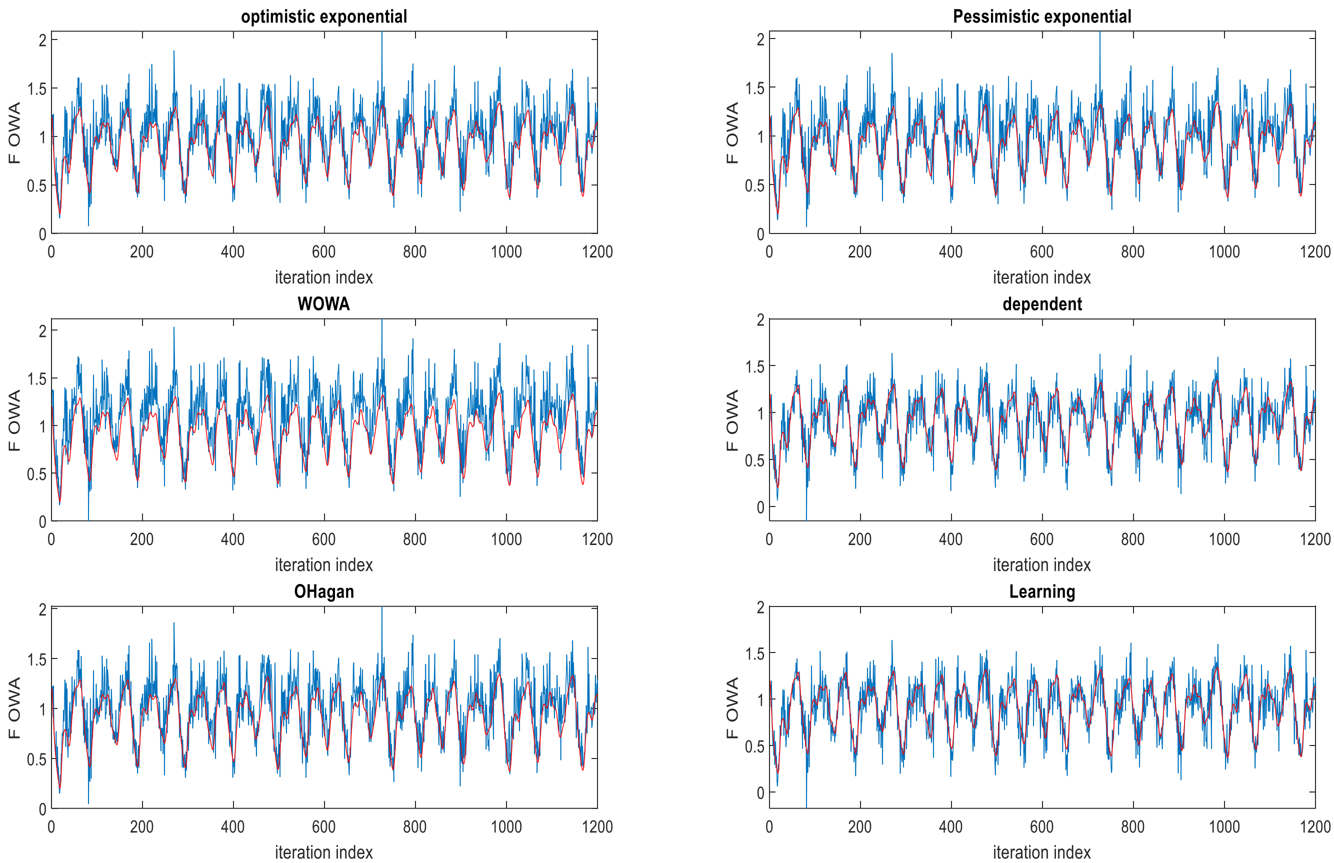
شکل 26 پارامترهای استخراج شده برای هر شش روش

و در نهایت خروجی را برای همه‌ی روش‌ها رسم نمودیم و بر روی شکل مقدار تخمین زده شده در هر روش را با مقدار واقعی مقایسه نمودیم. لازم به ذکر است که بدلیل اینکه در مسأله ذکر شده بود در انتها بدین صورت شکل‌ها را در کنار هم و در هر شکل دو نمودار تخمینی و واقعی قرار گیرد لذا قرار دادنِ legend موجب کم شدنِ کیفیتِ شکل می‌شد؛ به همین جهت لازم به ذکر است که رنگ آبی مقدار تخمینی را در هر روش نشان می‌دهد و رنگ قرمز نیز نماینده‌ی مقدارِ واقعی در هر یک از نمودار هاست.



شکل 27 مقایسه‌ی مقدار تخمینی و واقعی برای هر یک از شش روش

شکل زیر از کیفیت بیشتری برخوردار است. (رنگ آبی مقدارِ تخمینی در هر روش و رنگِ قرمز نماینده‌ی مقادیرِ واقعیِ اندازه‌گیری)



شکل 28 مقایسه‌ی مقدار تخمینی و واقعی برای هر یک از شش روش (تصویر با کیفیت بالاتر)

تحلیل

در هر بخش توضیحات مختصری آورده شد اما برای یک جمع‌بندیِ نهایی، یک تحلیل مناسب را به عنوانِ خاتمه‌ی کار ضمیمه می‌نماییم.

سوال از ما خواسته که یک بررسی بر روی خطاها انجام دهیم لذا در اول خطاها را برای هر روشِ اندازه‌گیریِ خطا مقایسه می‌کنیم.

: MSE

$$MSE_{Learning} < MSE_{dependent} < MSE_{pessimistic} < MSE_{OHagan} < MSE_{Optimistic} < MSE_{WOWA}$$

: RMSE

$$RMSE_{Learning} < RMSE_{dependent} < RMSE_{pessimistic} < RMSE_{OHagan} < RMSE_{Optimistic} < RMSE_{WOWA}$$

: MAE

$$MAE_{Learning} < MAE_{dependent} < MAE_{pessimistic} < MAE_{OHagan} < MAE_{Optimistic} < MAE_{WOWA}$$

همانطور که مشاهده می‌فرمائید، شاهدِ یک روندِ یکسان بینِ هر سه نوع خطا هستیم و لیلِ آن وجودِ ماهیتِ یکسان در روشِ محاسبه‌ی این سه خطاست. از طرفی کاملا واضح است که در روشِ learning، خطا تا حدِ قابلِ توجهی کاهش پیدا می‌کند. این میزان کاهش برای روشِ یادگیریِ وزن‌ها چیزی حدودِ 50 الی 60 درصد است. بعد از آن روشِ وابسته خطا را تا حدِ خوبی کاهش می‌دهد و دلیلِ آن به نظرِ من، مقداردهی به وزن‌ها برای هر مشاهده است که طبیعتا انعطاف پذیریری حل را بسیار بالا برده و خطا را تا حدِ خوبی کاهش خواهد داد. بعد از این دو روش، روشهای OHagan و خوشبینانه با اختلافِ بسیار کمی نسبت به هم، عملکردِ خوبی را در کاهشِ خطا دارند. این اختلاف در مدلسازیِ مسأله‌ی ما در اوردرِ هزارم برای هر یک از خطاهاست و می‌توان طبق مشاهداتِ این پروژه، کاراییِ این دو روش را در کاهشِ خطا معادلِ یکدیگر دانست. پس از این دو روشِ WOWA است که در کاهشِ خطا عملکردِ بدتری را نسبت به پنج روشِ ماقبل دارد. به نظر می‌رسد دلیلِ این امر را می‌توان مقدار دهیِ دلخواه برای پارامترهای اولیه دانست. در واقع در روشِ WOWA ما وزن‌ها را برای حالت اولیه دلخواه انتخاب کرده و همچنین p را. اگرچه در انتخاب این دو شروطی وجود دارد ولی محدودیت‌ها مانندِ سایرِ روش‌ها نیست. از طرفی fitting هم می‌تواند تاثیرگذار باشد. چراکه نوعِ fitting و محاسبه‌ی ضرایب باز به گونه‌ای به ورشِ ما برای فیت کردنِ منحنی برمی‌گردد. لذا دور از انتظار نیست که در این آزمایش به این نتیجه بر بخوریم که WOWA نسبت به پنج روشِ دیگر، از کاراییِ کمتری برای کاهشِ خطا برخوردار است.

در کل می‌توانیم نتیجه بگیریم که اُپراتورهای owa فوق از کارایی نسبتا خوبی برای کاهشِ خطا برخوردار بودند و روشِ learning نیز بهترین کارایی را در این شش روش برای کاهشِ خطا از خود بروز کرده بود. با این حال دیگر روش‌ها هم عملکردهای قابلِ قبولی را دارند و این از شکل 28 و جدولِ موجود در شکل 26 کاملا واضح است. با این حال برای یک جمع بندی به نظر می‌رسد روش learning برای حلِ مسائلِ owa می‌تواند گزینه‌ی خوبی باشد البته محدودیتی که این روش دارد، دانستنِ مقادیرِ واقعی از اندازه‌گیری است که در محاسبه‌ی لانداها به طور مستقیم تاثیر می‌گذارد. در واقع اگر مقادیر واقعیِ اندازه‌گیری را در اختیار نداشته باشیم ناگزیر باید روش‌های دیگری را برای ساخت خروجی برگزینیم.