El Tractorcito
Manual de Usuario

Ruben Chaer
Setiembre 2024 - Montevideo - Uruguay

1. Introducción

La aplicación Tractorcito fue desarrollada en la plataforma de Simulación de Sistemas de Energía Eléctrica SimSEE en el proyecto ANII-FSE_1_2017_1_144926 - "Planificación de inversiones con energías variables, restricciones de red y gestión de demanda" (2018-2020) Fondo Sectorial de Energía ANII.

El Tractorcito es un Robot, que mediante técnicas de aprendizaje por refuerzo (ensayo y error), aprende cómo operar un sistema de energía eléctrica. Este aprendizaje se expresa en lo se llamaremos una Política de Operación (PO) que es una mapeo (función) de la forma:

$$u = PO(X, r, k) \quad , \qquad (1)$$

donde u es el vector de control (potencias a despachar en cada máquina, operación de los vertederos, etc.), X es el vector de estado del sistema (volumen de agua en los embalses, estado de escurrimiento de las cuencas hidrológicas, etc.), r es el vector de información con las variables exógenas (precio del barril de petróleo, caudales afluentes, velocidad de viento, etc.) y k es el paso de tiempo de simulación.

El Tractorcito, en base a Simulaciones de Exploración, sobre una Sala SimSEE dada, va aprendiendo una PO buscando minimizar el costo esperado de la operación futura.

La solución clásica de SimSEE [1] sería primero realizar una etapa de optimización, aplicando el algoritmo recursivo de Bellman [2], para obtener una PO óptima y luego realizar simulaciones para observar resultados de simular con dicha PO.

El Tractorcito no utiliza la recursión de Bellman. Simplemente comienza con una PO NULA (es decir sin información) y en base a simulaciones va recabando la información relevante y aprendiendo una PO que debiera tender a una óptima. Este aprendizaje se realiza mediante un bucle de aprendizaje como se esquematiza en la Fig.1.

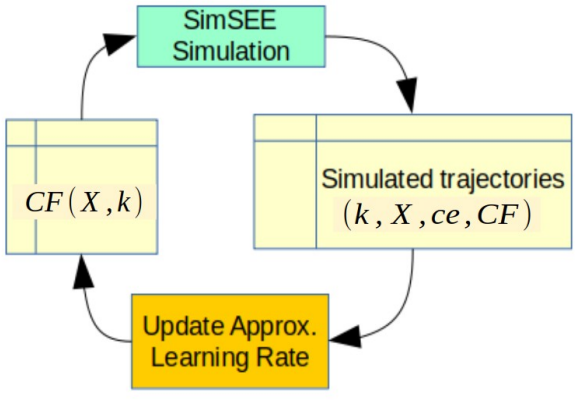


Fig. 1: Bucle de aprendizaje

La PO se almacena en una función que representa el valor esperado de la operación futura y que llamaremos función de Costo Futuro $CF(X,k)$. En esta función (más precisamente, en sus derivadas direccionales respecto a las variables de estado) se encuentra codificada la PO. Para obtener el vector de control en un paso k dado, conocido el estado X_k

al inicio del paso y las entradas no controlables r_k durante el paso, el vector de control se calcula minimizando el costo operar el sistema en el paso de tiempo (suma de los costos incurridos en los combustibles consumidos, costos de importación menos ingresos por exportación, etc.) más el valor de $CF(X',k+1)$ en el estado X' al que el sistema arribe al final del paso k .

La PO Nula con que inicia el aprendizaje el Tractorcito corresponde a considerar $CF(X,k)=0 \forall X,k$.

El despacho óptimo sería la solución al problema de optimización que minimiza:

$$u=\underset{u}{arg\,min}\{ce(X,r,u,k)+CF_h(X',k+1)\} \quad @X'=f(X,u,r,k)$$

ec.(2) Recursión de Bellman con entradas del paso conocidas.

Donde:

- X es el vector de estado del sistema al inicio del paso,
- r es el vector de entradas no controladas (exógenas),
- u es el vector de control (las potencias despachadas y demás variables de decisión del paso),
- k el ordinal que identifica el paso de tiempo al inicio de la resolución del paso,
- $ce(X,u,r,k)$ es el costo de etapa, incurrido en el paso de tiempo (suma de combustibles, importaciones, etc. menos ingresos por exportaciones)
- $f(X,r,u,k)$ la ecuación de evolución del estado
- X' es el estado al final de paso k impuesto por la ecuación de evolución de estado a partir del estado al inicio del paso, el vector de control, de entradas no controlables y del paso k .
- $CF_h(X,k)$ la función que representa el valor esperado del costo futuro de la operación con en la iteración de aprendizaje h .

La (2) es la base del cálculo de una Política Óptima de operación, partiendo de un futuro lejano en que se supone $CF_{PO}(X,+\infty)=0$ y se aplica la recursión viniendo paso a paso hacia el presente, resolviendo el problema de optimización para cada valor del vector de estado X . Esta solución fue propuesta por Richard Bellman en 1957[2] y en su publicación alertaba sobre lo que llamó la Maldición de la Dimensionalidad (hoy conocida como la Maldición de Bellman) que indica que en la medida en que crece la cantidad de variables de estado (componentes del vector X) el problema computacional de realizar el cálculo barriendo una discretización del espacio de estado es intratable. Observar que si cada dimensión del espacio de estado se discretiza en M posiciones de la variable y si la cantidad de variables de estado en N el barrido del espacio de estado implicaría resolver el problema para N^M posiciones del estado. Además, si se resuelve el problema para R realizaciones de las variables exógenas (proceso estocásticos en su mayoría) y se debe resolver el problema para K pasos de tiempo se tiene en total que resolver: $N^M \times R \times K$ veces.

La lucha contra la Maldición de Bellman es permanente cuando se trata de la operación óptima de sistemas dinámicos. El Tractorcito un arma más en esa batalla.

Desde la página del proyecto (https://simsee.org/investigacion/tractorcito.html) se puede acceder a la memoria final del proyecto y a videos explicativos de los fundamentos.

Hay dos aplicaciones de El Tractorcito, una con interfase gráfica que debe ser utilizada para configurar los parámetros de aprendizaje y puede también usarse para ejecutar el aprendizaje en un modo en que en cada iteración se visualiza gráficamente el comportamiento y otra aplicación sin interfase gráfica que permite ejecutar un aprendizaje en forma independiente del usuario utilizando una Sala y juego de parámetros dados.

1.1. Aprendiendo con mucho ruido

a) Efecto de la incertidumbre sobre la capacidad de aprendizaje

De las publicaciones y libros consultados, a mi entender, no hay un tratamiento adecuado del efecto de la incertidumbre impuesta por los procesos estocásticos sobre la capacidad de aprendizaje del Agente. La energía afluente hidráulica en los sistemas de energía eléctrica es un proceso estocástico que impone correlaciones temporales de más de un año de duración. En los sistemas energéticos de América Latina, donde la hidroelectricidad participa en más del 50% de las matrices de generación, esa variabilidad se traduce en una varianza importante en la función de valor del estado $CF(X,k)$. Como el aprendizaje pasa por la resolución del problema de despacho óptimo de un paso de tiempo (típicamente horario, diario o semanal según el horizonte de tiempo de análisis) minimizando la suma del costo incurrido en el paso (Costo de Etapa: ce) con el valor esperado de la función de valor $CF(X,k)$ evaluada en el estado de llegada del sistema al final del paso, la relación entre la variabilidad de la función de valor (Costo Futuro: CF) y la variabilidad del Costo de Etapa juega un papel importante. El problema de despacho puede ejemplificarse como si la función que da el valor esperado del costo futuro al final del paso de tiempo fuera la altura del horizonte (fin del paso de tiempo) y un caminante debe minimizar (para resolver el despacho del presenta paso de tiempo) el costo de caminar hasta el horizonte más la altura del lugar a donde llegue en dicho horizonte como se muestra en la Fig.2. La notación matemática del problema de despacho sería:

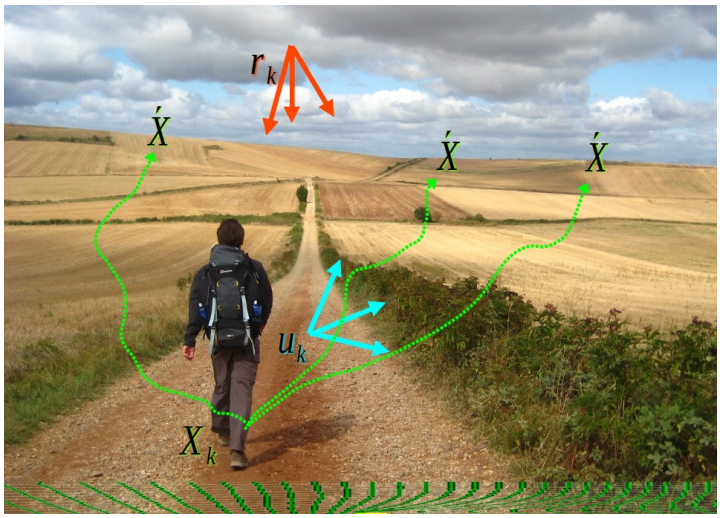


Fig. 2: Resolución de un paso de tiempo

$$\min_{u} \left\{ ce(X_k, u_k, r_k, k) + CF(X_s, k+1) \right\} \quad @ \begin{cases} u \in \Omega(X_k, r_k, k) \\ X_s = f(X_k, u_k, r_k, k) \end{cases} \quad ,(3)$$

Ahora bien, si bien $CF(X,k)$ representa el valor esperado de la operación futura, ese valor esperado es sobre una distribución de valores que tiene una variabilidad ordenes de maginitud superior a la variabilidad de ce. A modo de ejemplo, la Fig.3 muestra la distribución del valor presente del costo futuro de la operación para el sistema de generación de Uruguay a partir de un estado X dado. Como se puede apreciar, el valor esperado es del orden de 8200 MUS$, pero con una dispersión que hace que según "la suerte futura" ese valor pueda variar entre 8000 y 10000 MUS$. Por otro lado la Fig.5 muestra la distribución de ce para una simulación de paso semanal para el mismo sistema uruguayo. Como se puede apreciar, la dispersión va en tre 5 y

25 MUS$ lo que es muy inferior a los valores de dispersión del costo futuro. Como ambos valores se obtienen por simulaciones del tipo Montecarlo, la comparación que tiene que hacer el caminante de la Fig.2, se vuelve un poco difícil pues su horizonte presenta variaciones (ruido) muy superior a el costo de caminar hasta el horizonte como se ejemplifica en la Fig.

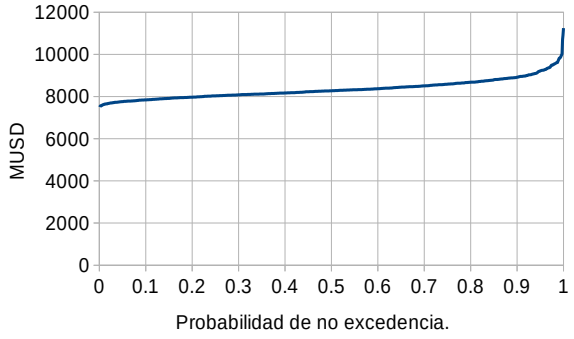


Fig. 3: Distribución de CF(X,k) para el sistema uruguayo

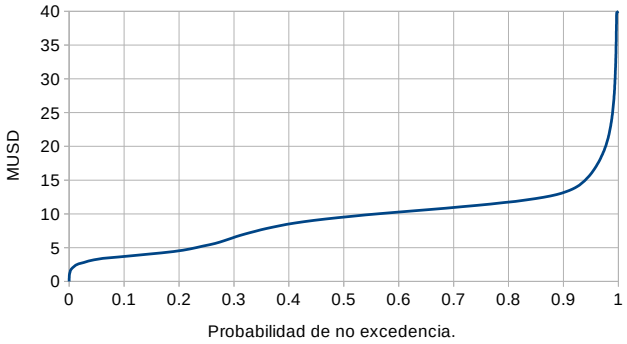


Fig. 4: Distribución del costo de etapa (semanal)

En las primeras implementaciones realizadas no se tuvo en consideración este aspecto y el resultado es que no se lograban políticas de operación que se acercaran a las obtenidas con la recursión clásica de Bellman. Una vez que se entendió la necesidad de controlar la varianza de la inforamción se realizó la implementaicón actual que utiliza la técnica conocida como Common Random Numbers (CRN) para lograr mitigar el efecto de la varianza de CF+CE sobre la capacidad de aprendizaje y con resultados que en las Salas de prueba muestran que se logra aprender políticas de operación que llevan a costos iguales a los obtenidos con la recursión clásica de Bellman.

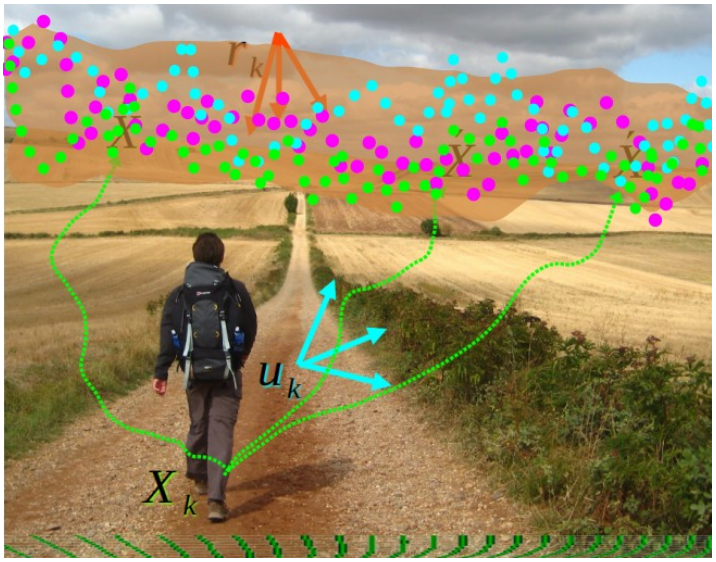


Fig. 5: Efecto de la incertidumbre futura sobre $CF(X,k)$

b) Aprender las diferencias espaciales de la función de Valor del Estado

En las publicaciones y libros consultados, no se identifica con claridad, que el objetivo no es una representación de CF sino de sus derivadas direccionales. El problema de la varianza del gradiente del beneficio (reward en la terminología de aprendizaje automático) y las ventajas de restar una "línea-base" se presenta en algunos trabajos como [3] y [16]. En dichos trabajos también se establece una diferencia entre "el beneficio inmediato" (el recibido en el paso de tiempo) y el gradiente del beneficio futuro (derivada de CF respecto de las variables de estado). Como ya se mencionó, en el caso de los sistemas de energía eléctrica, la varianza del costo del paso de tiempo (o costo de etapa) y la varianza del costo futuro en los estados posibles de llegada al final del paso

de tiempo (ver ejemplo de la Fig.3 y 4) son de ordenes de magnitud de diferencia lo que dificulta el aprendizaje. Cuanto menor sea el paso de simulación mayor será el cociente entre la varianza de $CF(X',k+1)$ (en un estado de llegada al final del paso) y del costo incurrido en el paso $ce(X_k,u_k,k)$. En los trabajos antes mencionados se utilizan técnicas de aprendizaje automático de las conocidas como model-free que implica que no se dispone de las ecuaciones del modelo y por tanto el aprendizaje es sobre la función de Valor-Acción: $Q(X,u,k)$ que representa el valor esperado de la operación futura a partir del estado X al inicio de paso k si se aplica el vector de control u y no de la función $CF(X,k)$. En el Tractorcito, se supone las ecuaciones del modelo conocidas (son los modelos disponibles de los generadores, centrales hidráulicas, etc.) y por eso el foco es en aprender la función $CF(X,k)$ y no $Q(X,u,k)$. La función $CF(X,k)$ es dimensionalmente mucho menor que $Q(X,u,k)$ dado que no contiene el vector de control u. Solo a modo de ejemplo, en simulaciones realizadas del conjunto Argentina, Brasil, Paraguay y Uruguay se consideraron 76 variables de estado y 5000 variables de control. Habiendo decidido aprender $CF(X,k)$ en lugar de $Q(X,u,k)$.

Como lo relevante para la resolución del problema de despacho (3) es la información de las derivadas espaciales de $CF(X,k)$, la solución implementada hace foco es que en crear un Agente que aprenda las diferencias espaciales de $CF(X,k)$ y no los valores de $CF(X,k)$ propiamente. Para reforzar que la información relevante está en las diferencias espaciales de la función $CF(X,k)$, observar que el problema (3) no cambia su solución si se suma un valor constante a la función de costo futuro. También observar que al caminante de la Fig.2, le importa minimizar el costo de caminar hasta el horizonte más la altura del horizonte al lugar donde llegue, lo cual hace que lo relevante sea la diferencia de altura entre los lugares a los que puede llegar.

2. Aplicación: tractorcito_GUI_rsr

Dada una Sala SimSEE, queda definido el Espacio de Estado de la Sala. Es decir se conoce el vector de estado X .

El primer paso en la configuración es entonces seleccionar la Sala. Para ello en abra la aplicación "tractorcito_GUI_rsr" y tendrá una pantalla como se muestra en la Fig.6. Como se puede observar, tiene solapas "Configurar", "Simular", "Monitor" y "(no usar, experimentos)".

Para la configuración se usan las solapas Configurar y Simular, la solapa Monitor le permite observar gráficamente la aproximación a la función de costo futuro que la estructura de redes neuronales que haya configurado. La solapa (no usar, experimentos) como su nombre lo indica no debe ser utilizada.

En la descripción de los parémetros se hace referencia a la solapa de la aplicación tractorcito_GUI en la los mismos son configurables.

2.1. Solapa Configurar

La Fig.6 muestra el formulario de la sola Configurar.

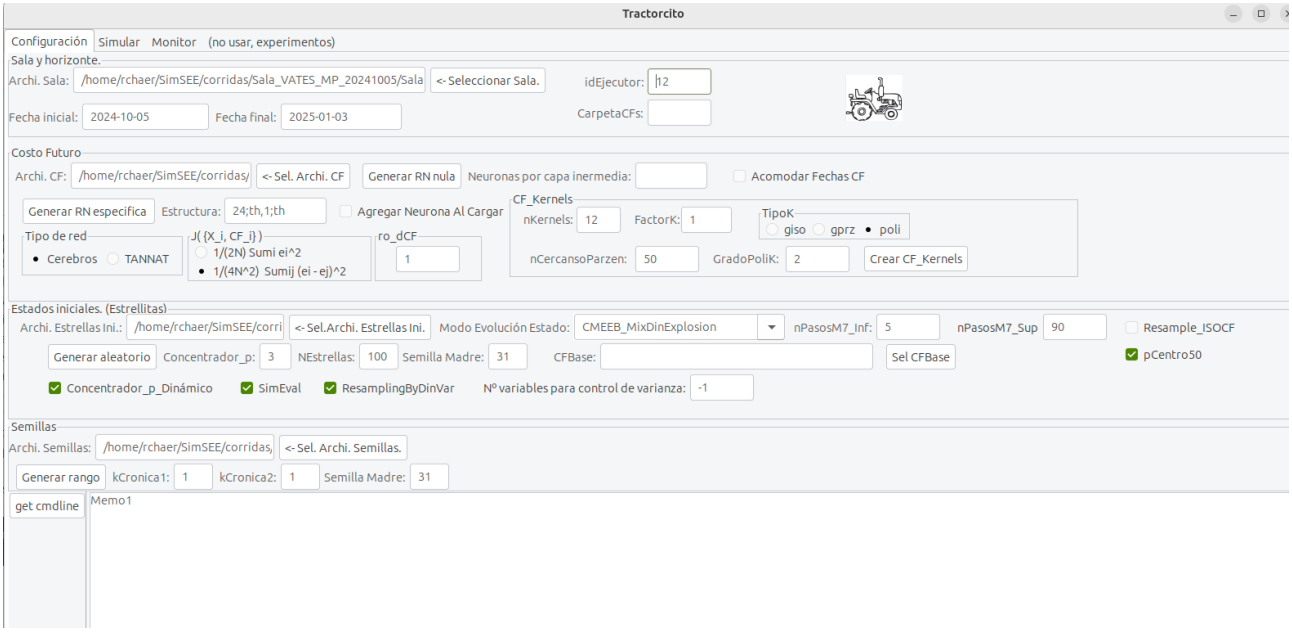


Fig. 6: Solapa Configuración

En esta solapa se selecciona la Sala y se configuran parámetros que definen el tipo de Red Neuronal y su estructura. También se configuran parámetros que determinan el conjunto de estados Iniciales (Estrellitas), las Semillas aleatorias para las simulaciones de exploración y parámetros que determinan las explosiones de estado para recuperar la capacidad de exploración.

El botón "<- Seleccionar Sala" le permite seleccionar la Sala SimSEE para la que desea entrenar una Política de Operación. Al seleccionar la sala se le completarán los casilleros Fecha Inicial y Fecha Final con los valores correspondientes a la Fecha de Inicio de Simulación y fin de

Optimización que se lean de la Sala. El aprendizaje de $CF(X,k)$ se realizará para ese horizonte temporal.

2.2. Solapa Simular

En la solapa Simular (ver Fig.7) se configuran los parámetros relacionados con la estrategia de ajuste de los parámetros de la Red Neuronal a los datos que arriban con cada simulación de exploración.

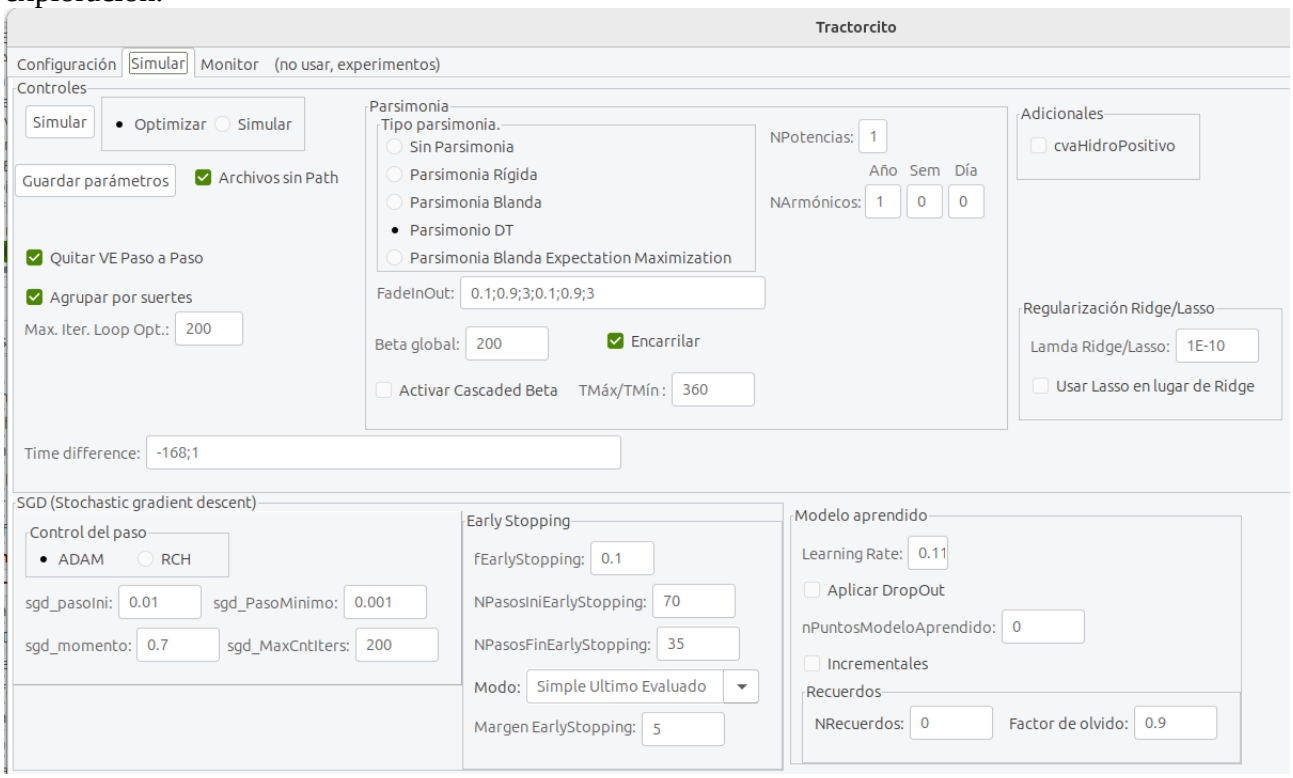


Fig. 7: Solapa Simular

El botón "Simular" permite lanzar directamente desde la aplicación tractorcito_GUI_rsr el bucle de aprendizaje. Inmediatamente a la derecha del botón hay un selector que le permite indicar si se lanza un aprendizaje completo o solo se debe realizar un simulación de exploración.

Si selecciona "Optimizar", se ejecutan etapas de aprendizaje consistentes en una simulación de exploración seguida de un ajuste de CF a los datos. La cantidad de etapas de aprendizaje a ejecutar se especifica en el casillero "Max.Iter.Loop Opt." (en el ejemplo de la Fig.7 se ejecutarán 200 iteraciones del bucle de aprendizaje).

Si se selecciona "Simular", solo se ejecuta la simulación de exploración. Esta opción se previó para facilitar la investigación de alternativas del ajuste de CF a datos (por ejemplo con aplicaciones externas programadas en otros lenguajes).

Al presionar el botón "Simular" se genera un archivo "params.prm" en la misma carpeta que la sala con el juego de parámetros completo. Este archivo junto con los archivos: "CF_RN_.xlt", "EstrellasIni.xlt", "Semillas.xlt" y la Sala (empaquetada con todos sus archivos) son los necesarios para la ejecución del Tractorcito.

Si solo quiere generar el archivo "params.prm", para su ejecución posterior con la aplicación tractorcito_rsr (sin la interface gráfica) presione el botón "Guardar parámetros".

Los casilleros "Quitar VE paso a paso" y "Agrupar por suertes" indican que la información de exploración debe ser tratada quitando el valor esperado de CF en cada paso temporal y que para el tratamiento se considere la información agrupada por igual suerte (semilla aleatoria).

El casillero "Time difference" le permite especificar el factor de descuento temporal y la cantidad de pasos temporales a acumular en la estimación de los valores de CF a partir de la información de las simulaciones de exploración.

En el panel "Parsimonia" se pueden configurar los parámetros relacionados con imponer continuidad temporal en la representación de CF (regularización de los parámetros respecto de su variación en el tiempo).

En el panel "Regularización Ridge/Lasso se puede configurar los parámetros relacionados con imponer regularización de los parámetros en general (no solo temporal).

Los paneles "Modelo aprendido" y "Recuerdos" permiten configura los parámetros relacionados con acumular la información de exploraciones anteriores.

El panel "SGD (Stochastic gradient descent) permite configurar los parámetros relacionados con el ajuste de los parámetros de las redes neuronales a los datos.

Por último el panel "Adicionales" se previó para incluir parámetros específicos del problema de despacho energético. Por ahora el único disponible es "cvaHidroPositivo" que, de marcarse, implica que se impondrá $-\frac{\partial}{\partial x_i} CF(X,k) > 0$ para aquellas componentes x_i del estado X que represente volúmenes de agua embalsada. Esta imposición se realiza independientemente de la información de las simulaciones de exploración. Imponer que el valor futuro del agua embalsada sea positivo puede facilitar/acelerar el aprendizaje pero en principio no sería necesario realizar dicha imposición.

2.3. Solapa Monitor

Luego luego de cada etapa de Simulación de Exploración viene una etapa de Ajuste del modelo $CF(X,k)$ a la nueva información.

En la solapa Monitor (ver Fig.8) se muestra gráficamente el ajuste. Para poder visualizar el ajuste (siendo el vector de estado de dimensiones generalmente > 1) se ordenan en cada paso de tiempo toda los puntos de información por orden decreciente de CF y se grafica el valor de CF contra el orden impuesto. Inicialmente, el eje

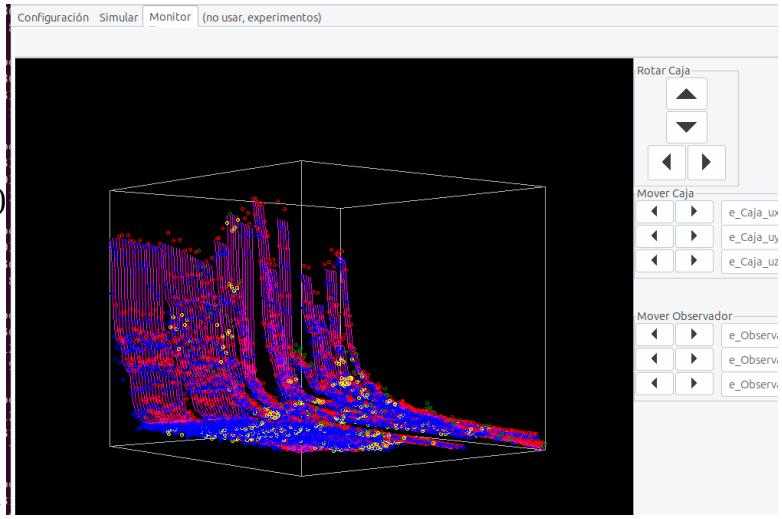


Fig. 8: Solapa Monitor

vertical representa el valor de CF y el horizontal la posición del punto de información en el orden decreciente de CF impuesto. Los diferentes pasos de tiempo se encuentran graficados "en profunidad" esto es en un eje perpendicular a la pantalla comenzando en la pantalla por el primer paso de tiempo y yendo hacia atrás de la pantalla en forma proporcional con cada paso de tiempo.

Para facilitar la visualización, se permite rotar la caja con las flechas del panel "Rotar Caja". En la Fig.8 se ha rotado un poco la caja del gráfico (usando la flecha que apunta a la izquierda en el panel "Rotar Caja") quedando así visible las diferentes curvas que representan la información de cada paso. Las curvas color fusia represetan los valores de la información de exploración. Los puntos Rojos y Azules representan los valores del modelo en los puntos de información usados para el ajuste. El color Rojo indica que el modelo da un valor más alto de CF que el informado en la muestra de exploración, el color Azul indica que el modelo estima un valor más bajo de CF que la muestra. Los puntos Amarillos corresponden a los puntos de información de exploración que excluidos de los datos de entrenamiento para el control de sobre-ajuste usando el mecanismo de EarlyStopping. El que en la figura parezcan puntos rojos y azules indica que el modelo está aproximando razonablemente a los datos. Si fueran todos rojos o todos azules indicaría que el modelo está sobreestimando o subestimando el valor de CF en todo el espacio.

3. Estructura de la Red Neuronal

En el casillero "Estructura:" debe especificar la estructura de la red neuronal a utilizar. En la figura aparece: "12;th, 1;th" que significa una primer capa de 12 neuronas del tipo tangente hiperbólica seguida de una capa (la de salida) con una neurona del tipo tangente hiperbólica. Como se está aprendiendo la función $CF(X,k):\Re^N \rightarrow \Re$, las neuronas de la primer capa tendrán $N=Dim(X)$ entradas y necesariamente la última capa especificada debe tener una sola neurona cuya salida será el valor de CF.

La cantidad de capas, neuronas por capa y tipo de neuronas a utilizar es una cuestión experimental. Cuanto más recursos de representación (más cantidad de capas y neuronas) se le de a El Tractorcito, mayor será la complejidad de su "Cerebro" y por tanto más difícil y prolongado su proceso de aprendizaje. Estructuras muy complejas no logran aprender a partir de un conjunto de observaciones (experiencias de vida) limitado. En resumen, no debiera suministrar más complejidad que la que los Operadores Reales son capaces de administrar. De nada le sirve tener más precisión que la que se tendrá en la operación real.

Presionando el botón "Generar RN específica" se crea una primer representación de $CF_0(X,k)$ con la estructura de Red Neuronal especificada. A esta primer representación es a la que llamaos la Política Nula pues cumple que $CF_0(X,k)=0$. Recordar que la información que determina la Política de Operación se encuentra en las derivadas direccionales $\frac{\partial}{\partial x_i}CF(X,k)$ y al ser $CF_0(X,k)=0$, se cumple para esa primer representación $\frac{\partial}{\partial x_i}CF_0(X,k)=0;\forall i$.

Antes de precionar el botón "Generar RN específica", hay que seleccionar el tipo de red (representación de $CF(X,k)$) se quiere utilizar. Por ahora las opciones disponibles son:

- Cerebros: Crea para cada paso de tiempo una red neuronal con la estructura indicada en "Estructura". Al crear una RN para cada paso de tiempo, la información temporal se encuentra capturada al cambiar la RN en uso según el paso de tiempo. Como se verá algunos parámetros permiten evitar variaciones abruptas entre los parámetros de las RN contiguas en el tiempo (imposición de parsimonia).
- TANNAT: (Traveler Artifitial Neural Network Around Time) que corresponde a una única red con la estructura especificada en "estructura" a la que se agregan a las entradas X un vector de entradas $y(k)$ que suministran la información temporal. Si se selecciona la estructura TANNAT. Se debe configurar en la solapa "Simular", los parámetros NPotencias y NArmónicos (Año, Sem, Día) que especifican las componentes a generar como entradas $y(k)$. A modo de ejemplo si se especifica NPotencias = 2, se generará una entrada con $y=k$ y otra con $y=k^2$ que le permitirán tener a la red una variación lineal y cuadrática con el tiempo en horizonte de análisis. En forma similar, los parámetros NArmónicos: Año, Sem, Día permiten especificar la cantidad de armónicos de ciclo anual, semanal y diario respectivamente se generaran para agregarse a las entradas de la red, permitiendo así que la misma pueda tener un comportamiento adaptado a los ciclos que se consideren importantes. Una vez especificados estos parámetros se debe presionar el botón "Generar RN Específica" de la solapa configuración para que se genere el archivo CF. Los parámetros FadeInOut también tienen efecto generando entradas que señalizan el inicio y final del horizonte de simulación. Esto tiene efecto sobre la adaptación de la función CF a la información del estado inicial y pronósticos al inicio del horizonte y para la adaptación a un enganche con otra CF (en caso de existir) al final del horizonte. Estos parámetros se tratan más en detalle en 10.1. c).

Al presionar el botón "Generar RN específica", se genera en la misma carpeta del archivo de la Sala seleccionado el archivo: CF_RN_.xlt (es un archivo de texto plano, con valores separados por tabuladores) que contiene la estructura de la RN seleccionada con los parámetros fijados para que sea la política nula.

Este archivo es pasado a formato binario al iniciar el proceso de entrenamiento y guardado en la carpeta en que se guardan las políticas resultantes con el nombre CF_0.bin y contiene la Política Nula que es la primera utilizada para comenzar el entrenamiento.

3.1. Tipos de Neuronas

El tipo de neuronas más comunes se pueden representar como una función $s= g(y):\Re \rightarrow \Re$ siendo $y=\rho_0+\sum_i \rho_i e_i$ una combinación lineal de las entradas a la que se suma un offset (bias) ρ_0 y $\{\rho_0, \rho_1, ... \rho_N\}$ los parámetros de entrenamiento.

Los tipos de neuronas actualmente admisibles son:

- 'li': Lineal. $s=y$
- 'th': Tangente hiperbólica. $s=\tanh(y)=\frac{e^y-e^{-y}}{e^y+e^{-y}}$
- 'exp': Exponencial. $s=e^y$
- 'sig': Sigmoide. $s=\tanh(y)=\frac{1}{1+e^{-y}}$

- 'lo': Logica. $s=y>0$
- 'relu': $s=(y>0)\,y$
- 'max': Máximo. $s=max(\rho_i e_i)$
- 'psmax': PonderSoftmax: $s=\sum_i \left(\frac{e^{\rho_i x_i}}{\sum_j e^{\rho_j x_j}} \right) x_i$

4. Semillas aleatorias

Una de las principales conclusiones del proyecto de investigación que dio lugar a El Tractorcito, es que la variabilidad de los procesos estocásticos involucrados futuros, es de tal magnitud, que hace difícil compara las variaciones sobre el valor esperado de la operación futura asociado a decisiones del presente sobre un paso de tiempo. Dicho de otra forma, el costo de un paso de tiempo (suma de combustible, racionamiento, importaciones menos exportaciones) es ordenes de magnitud inferior a la variación esperada del costo futuro en un estado dado y por tanto dificulta la estimación de $\frac{\partial}{\partial x_i} CF(X,k)$. Para lograr estimar mejor las variaciones de $CF(X_a,k+1)-CF(X_b,k+1)$ entre dos estados de llegada al final del paso k se vuelve imperioso utilizar técnicas de reducción de varianza. Como las simulaciones realizadas en SimSEE son del tipo de Montecarlo, lo que se utiliza es la técnica conocida como Common Random Numbers, que implica que para las comparaciones se somete el futuro a la misma "suerte". Esto implica someter el sistema a la misma realizaciones de los procesos estocásticos para las trayectorias simuladas a partir de los estados $(X_a,k+1)$ y $(X_b,k+1)$. En el panel "Semillas" dentro del panel "Configuración" se especifica la Semilla Madre (31 por defecto) aleatoria con que se inicializa el generador de números aleatorios y en los casilleros kCronica1 y kCronica2, qué realizaciones e las generadas a partir de esa Semilla Madre se quieren simular como realizaciones iniciales del proceso de aprendizaje. Por defecto ambos valores están en 1, indicando que se generará una única realización.

Presionando el botón "Generar Rango" del panel "Semillas" se genera el archivo de texto plano: Semillas.xlt, con las semillas a utilizar.

5. Estados iniciales y modos de evolución de estado

5.1. Estrellitas iniciales

En el panel "Estados Iniciales" (Estrellitas) se especifica cómo se considerará el estado inicial del sistema para el proceso de exploración (simulación de exploración) que permite recolectar información para mejorar la aproximación $CF(X,k)$. En el parámetro NEstrellas se especifica la cantidad de estados iniciales a considerar (por defecto 100). Llamamos Estrellitas a un conjunto de estados. Los estados iniciales se generan en forma aleatoria, usando la Semilla Madre especificada en el mismo panel. Para la generación de cada estrella se considera el estado inicial especificado en la Sala y se lo altera sortenado en cada dimensión una perturbación. En cada variable de estado la perturbación se realiza con probabilidad 50% aumentando el valor y con probabilidad 50% reduciendo el valor:

$$x_{ij}=x_{i0}+(w_{ij}>0.5)(\hat{x}_i-x_{i0})u_{ij}^p-(w_{ij}\le 0.5)(x_{i0}-\check{x}_i)u_{ij}^p \quad ,(4)$$

Donde i identifica la componente del vector de estado, j la estrellita que se está generando y w_{ij} y u_{ij} dos números aleatorios generados con distribución uniforme. El proceso de generación es entonces, para cada estrella a generar j, para cada una de las componentes i del estado se aplica (4) y así se generan las posiciones iniciales del estado.

En el campo "Concentrado_p" se debe especificar el valor a utilizar de p. Los valores aconsejables son $1 \le p \le 3$. A mayor valor de p se está aumentando la capacidad de exploración del espacio de estado, pero alejando las trayectorias del "curso más probable" lo que requerirá una mayor cantidad de bucles de aprendizaje. En caso de marcar el casillero "Concentrador_p_dinámico" se comienza aplicando $p=1$ y se tiende durante las iteraciones al valor especificado en "Concentrador_p". Si el casillero está desmarcado se aplica desde el principio el valor especificado en "Concentrador_p".

La fórmula aplicada para la variación dinámica del concentrador p es:

$$p-(p-1)e^{-k/50} \quad , \qquad (5)$$

siendo $k=0,1,\ldots$ la iteración de exploración del bucle de aprendizaje. La idea es comenzar explorando priorizando la exploración no guiada por las trayectorias y en la medida en que se avanza en aprendizaje concentrar la exploración en las regiones cercanas a las trayectorias más probables.

Presionando el botón "Generar aleatorio" del panel "Estados Iniciales (Estrellitas)" se genera el archivo de texto plano: EstrellasIni.xlt con las posiciones iniciales del estado a utilizar para las simulaciones.

Para las simulaciones de exploración se utiliza el producto carteciano de las semillas del archivo Semillas.xlt y de las posiciones de estado del archivo EstreallasIni.xlt. De esta forma, cada estado inicial es simulado con la misma realización de los procesos estocásticos futuros.

5.2. Modos de evolución de estado

Observar, que saliendo de estados diferentes, las crónicas asociadas a la misma realización de los procesos estocásticos irán convergiendo (por características del sistema, como los límites de almacenamiento en los lagos y rendimientos de conversión de energía). Esto implica que si miramos el conjunto de trayectorias que se generan por el conjunto de estados sometidos a una misma realización de los procesos estocásticos, esas trayectorias irán convergiendo, reduciendo la distancia entre los estados y por tanto perdiendo la capacidad de exploración de la función $CF(X,k)$. Esta convergencia está ocasionada por la necesidad de utilizar la técnica de Common Random Numbers para controlar la varianza. Para recuperar capacidad de exploración, el conjunto de estrellitas se "explota", separándolas nuevamente. Esta explosión de los estados puede controlarse con algunos parámetros.

El parámetro Modo Evolución Estado permite seleccionar entre 11 estrategias de manejo de la evolución de los estados (ver secc. 14). Por defecto está seleccionado

CMEEB_MixDinExplosion que implica que la simulación se realizará combinando la evolución dada por Dinámica del sistema (esto es la evolución natural dada por la ecuación de evolución de estados) con la Explosión del conjunto de estrellitas. El modo CMEE1_Cilindro, mantiene constantes las estrellitas (no evolucionan con la ecuación de estados). La mayoría de los modos corresponden al proceso de investigación y el modo recomendado es el marcado por defecto. El modo CMEE7_Dinamica_submodo, corresponde a usar la dinámica natural del sistema en forma permanente.

a) Cantidad de pasos mínimo y máximos con dinámica del sistema

Los parámetros nPasosM7_inf y nPasosM7_Sup, imponen mínimo y máximo de transiciones de estado en modo M7 (evolución natural del estado), en el caso de que se haya seleccionado alguna de las modalidadas que implican mezclar explosiones del estado con evolución natural. Por defecto los valores son: nPasosM7_inf = 5 y nPasosM7_Sup = 90. Esto implica que cuando se resuelva explosionar los estados se va a respetar que por lo menos se realizarán 5 transiciones consecutivas con la evolución natural del sistema y un máximo de 90.

b) Explosión de estados con muestreo uniforme de CF

Los casilleros Resample_ISOCF y pCentro50 regulan cómo se realiza la explosión de estados. Si se marca Resample_ISOCF, la explosión se intenta realizar de forma de distribuir las estrellitas en forma de concentrarlas más en los lugares de mayor variación del costo futuro y menos en las de menor variación. Si se marca el casillero pCentro50, cada estrellita se explota considerando la posición actual como la de probabilidad 50% (con el procedimiento usado para la explosión de los estados iniciales).

c) Control dinámico de la explosión de estados

Si los casilleros "ResamplingByDinVar" y "SimEval" están marcados se realiza una simulación de la Sala, a partir del estado inicial de la sala con la semilla aleatoria indicada en la sala un con la cantidad de crónicas indicadas en la sala para medir la varianza esperada en cada una de las variables de estado. Esa información es usada luego para que cuando la varianza de los estados en las simulaciones de exploración del aprendizaje (a partir de las estrellas iniciales) se reduce a un 0.7 de la varianza de la simulación de evaluación se produce la explosión de estados. Siempre respetando los parámetros nPasosM7_inf y nPasosM7_Sup. Este modo de explotación de estados está en etapa experimental. Por eso por defecto se deja desmarcado ResamplingByDinVar. En caso de marcarse, el parámetro: "Nº variables para control de varianza:" determina cuántas dimensiones del espacio de estado serán las controladas para detección de necesidad de una explosión de estados. El valor "-1" indica que se controlen todas las variables. Si se especifica por ejemplo 2, se estará indicando que se seleccionarán las dos variables de mayor inercia (detectada a partir de la variación que tienen en las simulaciones) y esas serán las utilizadas para detectar la necesidad de explotar las estrellas para recuperar capacidad de exploración. La idea de acotar las variables sobre las que se controla la pérdida de exploración es porque al no limitarse la incorporación de variables de estado, se agregan muchas de muy poca inercia y cuyo valor no tiene importancia en la determinación de la una Política de Operación (operable por el operador). A modo de ejemplo si se arma un sistema con 100 embalses y 300 baterías, difícilmente importe realmente el estado de los 400 reservorios. Y

muchos de ellos podrán tener variaciones rápidas y lo que importa al final es la suma de energía almacenada en el conjunto de reservorios de los que se puede disponer de la energía sin restricciones adicionales (como congestiones de generación). Por ese motivo, observar el conjunto de variables más pesadas limitado a un número similar a la cantidad de neuronas de la primer capa, parece ser razonable. Como ya se mencionó esta estrategia está en modo experimental.

5.3. Simulaciones de evaluación

El casillero SimEval, si se marca implica que para cada política de operación CF_{k}.bin guardad en una subcarpeta con el nombre de la Sala y el caso, se realiza una simulación, con el estado inicial, semilla y cantidad de crónicas indicadas en la sala (simulación de evaluación) y se calcula el valor esperado del costo total de operación en el horizonte de aprendizaje resultante de la operación con esa política CF_{k}.bin. Luego de esa simulación vendrá una simulación de exploración (desde las estrellitas se estado iniciales y con la estrategia de evolución y explosión de estados que se seleccione) y luego una etapa de entrenamiento de un nuevo CF que resultará en el archivo CF_{k+1}.bin. Los resultados de evaluación se van guardando en el archivo ve_cf0.xlt

La Fig.9, muestra un ejemplo de salida de archivo ve_cf0.xlt. El archivo es de texto plano, separado por tabuladores por lo que es fácilmente visualizable en LibreCalc o Excel.

cntIter_Global	mee	try_VE_CF_MUSD	sim_VE_CF_MUSD	SimEval_VE_CF_MUSD
0	11	0,0	0,0	357,7
1	11	378,6	381,0	354,8
2	11	397,0	398,7	349,6
3	11	401,6	403,5	348,2
3	11	0,0	0,0	348,2
4	11	382,3	384,5	349,8
5	11	394,5	396,9	346,8

Fig. 9: Archivo con registro de evaluaciones.

La primer columna "cntIter_Global" contiene el ordinal de la iteración de aprendizaje. Como el aprendizaje puede ser interrumpido y reiniciado, el contador de iteraciones tiene el adjetivo "Global" para diferenciarlo de las iteraciones del proceso de aprendizaje en curso. La aplicación de algunos parámetros del aprendizaje dependen de este contador de iteraciones. La segunda columna "mee" indica el modo de simulación usado durante la simulación de exploración. El valor 11 corresponde "CMEEB_MixDinExplosion". Este es el modo más usado actualmente.

Observar que hacia el final de la Fig.9 aparece repetida dos veces cntIter_Global = 3. La primer aparición corresponde a la última evaluación de una primer etapa de aprendizaje que fue interrumpida (interrumpiendo la aplicación tractorcito, intencionalmente o no). La segunda evaluación, corresponde al arranque de una segunda etapa de aprendizaje, lanzando de nuevo la aplicación tractorcito con el mismo juego de parámetros y reiniciando el aprendizaje a partir de la secuencia del último archivo CF_{k}.bin guardado de la etapa anterior. De esa segunda aparición solo es válida la última columna que tiene el Costo Futuro resultante de la simulación de evaluación con la política contenida en CF_19.bin. Como se puede apreciar coincide con el valor de la fila anterior (indicando que no se cambió ni la Sala, ni ningún parámetro que tenga influencia sobre la simulación de evaluación). La última columna "SimEval_VE_CF_MUSD" corresponde a la estimación del Valor Esperdo del Costo Futuro en Millones de dólares resultantes de la simulación de evaluación de la política CF_{k}.bin. Como se puede apreciar, en el caso de la Fig.9 ese valor fue decreciendo indicando que cada política obtenida fue mejor que la anterior. Esto no siempre es así, durante el aprendizaje se producen subidas y bajadas del valor "SimEval_VE_CF_MUSD" indicando que la información que arriba en cada iteración del bucle de aprendizaje no siempre colabora en obtener una mejor política.

Las columna "try_VE_CF_MUSD" representa el promedio de los costos directos de cada paso de simulación de las simulaciones de exploración, más el Costo Futuro al final del último paso de simulación (que puede ser diferente de cero si la Sala engancha con el Costo Futuro de otra Sala). Este valor tiene sentido si el modo de evolución de estado con que se realiza la exploración es el CMEE7_Dinamica_submodo pues solamente en ese caso, la concatenación de los pasos representa una trayectoria real del sistema. Este valor es obtenido sumando los valores publicados por la rutina que realiza la simulación de exploración.

La columna "sim_VE_CF_MUSD" corresponde al valor esperado del costo futuro devuelto por la subrutina que realiza la simulación de exploración. Nuevamente solo representa un costo real si el modo de exploración fuera el CMEE7_Dinamica_submodo.

Si no se marcó el casillero "SimEval", la última columna "SimEval_VE_CF_MUSD" contendrá un valor nulo. En ese caso, como un proxy del comportamiento del aprendizaje se tienen solamente los valores "try_VE_CF_MUSD" y "sim_VE_CF_MUSD", sabiendo que los mismos no

representan un valor real del costo esperado de la operación futura salvo se seleccione el modo de evolución CMEE7_Dinamica_submodo.

5.4. Función a aproximar J(X_i, CF_i)

La función de error para el entrenamiento de la RN, en caso de querer aproximar directamente el costo futuro sería:

$$J=\frac{1}{2N}\sum_{k,i}\left(CF_{ki}-M\left(X_{ki},k\right)\right)^2 \quad , \qquad (6)$$

Donde $M\left(X_{ki},k\right)$ es la salida de la RN cuando a su entrada tiene el vector de estado X_{ki} y se está al inicio del paso de tiempo k para la trayectoria de simulación i .

Si se marca el casillero "Quitar Valor Esperado Paso a Paso", luego de realizadas las simulaciones de exploración y reconstruida una estimación de entrenamiento $CF\left(X,k\right)$, antes de proceder al entrenamiento, para cada grupo de puntos $\left(X_i,CF_i\right)$ correspondientes al mismo paso de tiempo y misma semilla de simulación, se les quita el promedio de CF del grupo. De esta forma se elimina la información del offset que podría tener CF y se deja solo la información de las diferencias entre los puntos. Este tratamiento previo, facilita el entrenamiento si la función a aproximar es (6) eliminando parte del ruido asociado a la dispersión futura asociada a las realizaciones de los procesos estocásticos.

Como lo relevante a la PO son las derivadas direccionales $\frac{\partial}{\partial x_i}CF\left(X,k\right)$, para cada componente x_i del vector de estado X , como función de costo para el ajuste del modelo, más que (6) correspondería usarse:

$$J=\frac{1}{4N^2}\sum_{k,i,j}\left(\left(CF_{ki}-CF_{kj}\right)-\left(M\left(X_{ki},k\right)-M\left(X_{kj},k\right)\right)\right)^2 \quad , \qquad (7)$$

Siendo N la cantidad de trayectorias de información, $i=1,...N$ y $j=1,...N$ identifican un resultado (punto de información) en cada paso de tiempo.

Si se utiliza (6) o (7) para el ajuste del modelo se determina con el parámetro "modo-diff". Si modo-diff = 1 (valor por defecto) se aplica (7) y si modo-diff=0 se aplica (6).

6. Archivo params.prm

Con la aplicación tractorcito_GUI_rsr es posible configurar el Tractorcito. Como resultado, al presionar el botón "Simular" o el botón "Guardar parámetros" se crea el archivo params.prm con el listado completo de parámetros. Este archivo es el que puede pasarse a la aplicación

tractorcito_rsr para ejecutar sin interfase gráfica (por ej. en un cluster de computadoras). El archivo tiene referencia a archivos externos como el de la sala, las etrellitas, las semillas y la red neuronal con la política nula. Si la aplicación tractorcito_rsr va a ser ejecutada en otra computadora en una carpeta diferente a la de configuración se debe marcar "Archivos Sin Path" (que es el valor por defecto).

A continuación se listan los parámetros por orden de alfabético.

6.1. addneurons=0

Este parámetro indica la cantidad de neuronas a agregar en caso de estar reiniciando un entrenamiento. Se agregan tantas neuronas como se especifiquen en la primer capa de la red neuronal. Las neuronas agregadas son del mismo tipo de de las ya existentes. Este parámetro permite, luego de tener una política de operación entrenada (un CF_{nnn}.bin), cortar el entrenamiento, fijar este parámetro en la cantidad de neuronas a agregar (la última CF_{nnn}.bin) del entrenamiento cortado generando una nueva red neuronal (con las nuevas neuronas iniciadas con sus parámetros en CERO y el resto con los parámetros iniciales) y entrenando a partir de las simulaciones (iniciadas con la CF_{nnn}.bin) generar los sucesivos CF_{nnn+k}.bin

De esta forma se puede verificar si agregar complejidad a la primer capa de la red neuronal mejora o no la política de operación.

6.2. archisemillas=Semillas.xlt

Especifica el archivo con las semillas aleatorias.

6.3. arrnpci=12;th,1;th

Estructura de la Red Neuronal inicial usada como Política Nula. El formato es especificar las capas, separadas por "," (coma) y cada capa se especifica dando el número de neuronas seguido de un ";" (punto y coma) y del tipo de neurona.

En el ejemplo del título, la red será de dos capas. La primera con 12 neuronas del tipo tangente hiperbólica y la segunda (la capa de salida) con una única neurona del tipo tangente hiperbólica. La última capa debe tener siempre una sola neurona, dado que su salida será el valor de la función $CF\left(X,k\right)$.

6.4. beta_global=200

El parámetro beta_global corresponde al peso β de las ecuaciones anteriores.

La aplicación del peso de regularización es variable durante el entrenamiento. El parámetro beta_global especifica el valor al que tiende el valor β_k en las sucesivas iteraciones de aprendizaje k .

Dependiendo de la estructura de red neuronal, el parámetro β_k tiene el siguiente comportamiento:

// con esto intento que al inicio la información se propague más rápido y luego

// intente ajustar los detalles de variación más rápida

a) Red del tipo TAdminEstadosTANNAT

beta_global_efectivo := beta_global * (1 + exp(-cntIter_Global / 10))

$$\beta_k = \beta\left(1+e^{-k/10}\right) \quad , \qquad (8)$$

b) Red del tipo TAdminEstadosCerebros

$$\beta_k = \beta\left(1+10\,e^{-k/10}\right) \quad , \qquad (9)$$

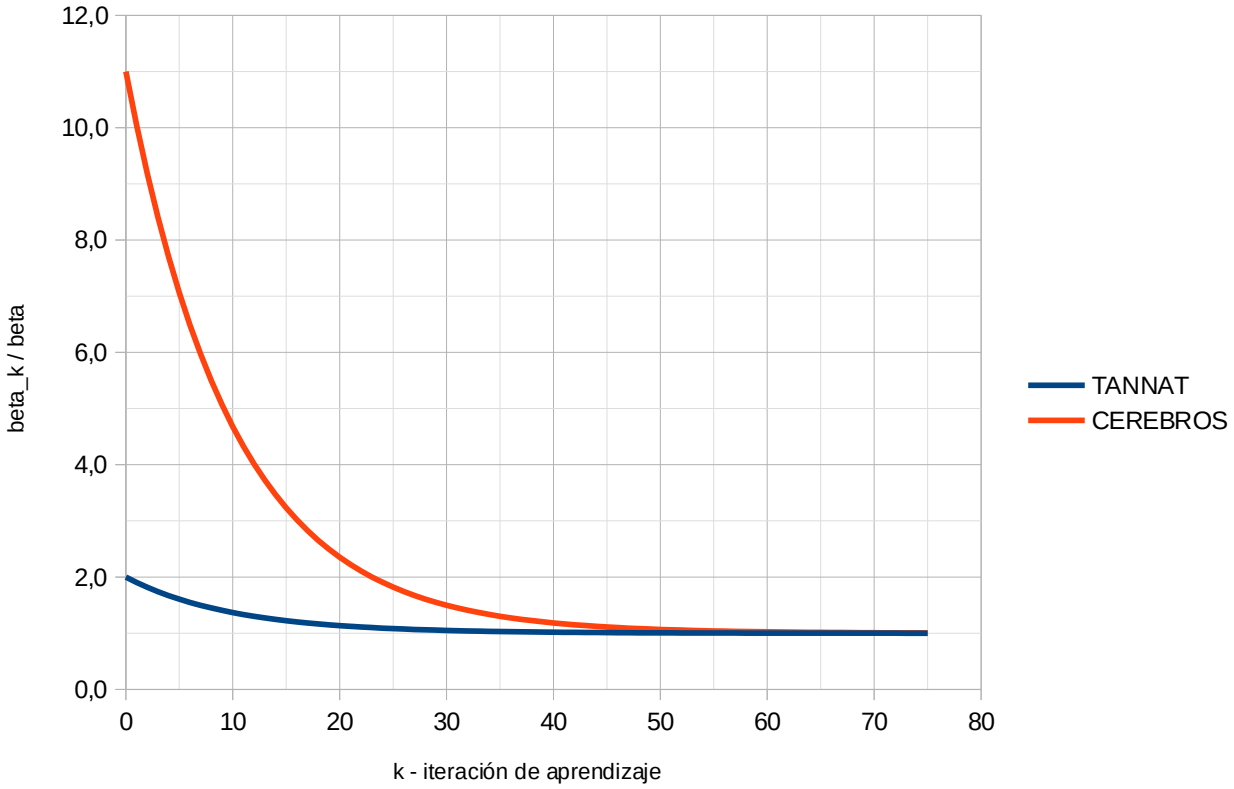


Fig. 10: Variación del parámetro beta_k a lo largo del aprendizaje

6.5. carpetacfs=

Este parámetro lo utiliza el optimizador de inversiones OddFace para indicarle a el Tractorcito en qué carpeta debe guardar los archivos con las aproximaciones obtenidas de CF .

6.6. cf=CF_RN_.xlt

Archivo de costo futuro con la Política Nula usado como inicio de las iteraciones de aprendizaje.

6.7. Cfbase=

En caso de especificarse, se lee al inicio y se intenta pasar el aprendizaje de ese archivo a la estructura especificada en el problema actual que llamaremos CFAprendiz.
Para realizar la transferencia de conocimiento, se muestrea CFBase en NPuntos (actualmente fijos en 10000) sorteados según la descripción del estado de CFAprendiz.

Si AgregarNNeuronas > 0 se agrega esa cantidad de neuronas a la Capa CERO de todos los cerebros si CFAPrendiz es del tipo TADminEstadosCerebros.
Ajusta los parámetros de CFAPrendiz paso a paso de tiempo y retorna el máximo (en los pasos) del error cuadrático entre la aproximación del APrendiz y la información de ajuste muestreada del Tutor.
Por ahora solo admite que CFAPrendiz sea del tipo TAdminEstadoCerebro

6.8. concentrador_p=3

Controla la dispersión de la explosión de estados que se produce durante las simulaciones de exploración dependiendo del modo de evolución de estado. Ver sección 5.2 para más detalles.

6.9. cvahidropositivo=0

Impone el valor del agua positivo de las centrales hidráulicas con embalse.

6.10. Dtfin="2024-09-15 08:00"

Fecha de fin del horizonte de aprendizaje.

6.11. Dtini="2024-09-05 08:00"

Fecha de fin del horizonte de aprendisaje.

6.12. estrellas=EstrellasIni.xlt

Archivo con las posiciones del estado al inicio de las simulaciones de exploración.

6.13. Fearlystopping=0.1

Factor de Early Stopping. Indica la proporción de las trayectorias de exploración que serán usadas para controlar el ajuste de CF en base a las trayectorias (1-fEarlyStopping).

6.14. flg_acomodarfechascf=0

Si flg_acomodarfechascf=1, las fechas de inicio y fin del costo futuro se ajustan acordes al horizonte de optimización de la sala. Esto está pensado para re-utilizar un CF entrenado con una

Sala cuyo horizonte estaba desfasado respecto de la Sala actual. Por ejemplo, en una Sala de paso horario con horizonte de optimización 168 horas, pensada para la programación del despacho semanal. Cuando el tiempo real avanza y se ajusta la sala para una nueva programación, en lugar de comenzar el entrenamiento desde cero, puede utilizarse este parámetro para indicar que se ajuste el CF aprendido con la Sala anterior. Es de esperar que la Pólitica de Operación no cambie radicalmente entre la programación de un día y la del día siguiente.

6.15. flg_activarcascadedbeta=0

Si flg_activarcascadedbeta=0 el parámetro β fija la parsimonia temporal con igual peso para todas las neuronas. Si flg_activarcascadedbeta=1 se cacula un peso β_i diferente para cada neurona de la capa de entrada. El parámetro ratio_tamx_tamin (ver 6.53) regula la variación del peso entre las neuronas.

$$\beta_i = e^{-\alpha \frac{i-1}{N_N - 1}} \quad , \qquad (10)$$

Siendo $i = 1, 2, \ldots N_N$ el índice que identifica las neuronas de la primer capa y $\alpha = \ln(\text{ratio\_Tmax\_Tmin})$ el logaritmo natural del parámetro ratio_Tmax_Tmin.

6.16. flg_agruparporsuertes=1

Este parámetro en 1 (valor por defecto) implica que el tratamiento de la información se realiza agrupando la misma por igual suerte (semilla aleatoria). De esta forma, si se ajusta por diferencias la función de costo futuro, o se quita el valor promedio de los costos CF_{ki} se realizará el ajuste sobre los conjuntos de punto de información correspondientes a la misma semilla. Recordar que la información corresponde a un conjunto de trayectorias de simulación desde un conjunto de estodos iniciales (ver 5.1) y para cada una de las semillas aleatorias (ver 4) especificadas para la exploración.

Si el parámetro flg_agruparporsuertes = 0 no se agrupan las muestras de las mismas semillas y se tratan todas en conjunto.

6.17. flg_concentrador_p_dinamico=1

El parámetro contentrador_p (ver 6.8) determina el grado de concentración de las explosiones de estado realizadas para recuperar la capacidad de exploración. Este parámetro regula si el concentrador_p se utiliza con su valor especificad fijo o si por el contrario se debe variar en función de la cantidad de iteraciones del bucle de aprendizaje.

Si flg_concentrador_p_dinamico= 1 se calcula, en cada iteración el concentrador_p efectivo a utlizar como:

$$cpe = \text{Concentrador\_p} \ . \ q_{Horizonte} + (1 - q_{Horizonte}) \quad , (11)$$

donde el coeficiente $q_{Horizonte}$ se calcula para cada iteración de aprendizaje $i_a = 1, 2, \ldots$ como:

$$q_{Horizonte} := \left(1.0 - e^{-i_a/50}\right) \quad , \qquad (12)$$

La idea es que el concentrador_p efectivo comience en 1 lo que hace que las explosiones sean no concentradas y entonces se explora con igual probabilidad cualquier región del espacio y en la medida en que el aprendizaje avanza, el concentrador_p efectivo tienda al valor especificado, priorizando así mantener la exploración cercana a las trayectorias resultantes de la simulación aplicando la Política de Operación que se ha aprendido.

6.18. flg_dropout=0

Si este parámetro se especifica en 1, en cada iteración de aprendizaje, antes de iniciar el ajuste del modelo, se selecciona al azar una neurona cualquier, excluyendo la de salida de la última capa y se ponen todos sus parámetros en CERO. Esto equivale a decirle a esa neurona que se olvide de todo lo aprendido y comience de nuevo.

Si flg_dropout=0, no se aplica este mecanismo

6.19. flg_encarrilar=1

Este parámetro es aplicable si se seleccionó el tipo de read nueronal TAdminEstadosCerebro (ver 9). La idea es que si flg_encarrilar=1, en cada etapa de aprendizaje, luego de haber ajustado la red neuronal con la nueva información, se determina la diferencia en los parámetros de la red neuronal de cada paso de tiempo respecto del anterior y a la red que tenga mayor discrepancia con sus vecinos se le impone el promedio de sus vecinos. Este mecanismo impone una continuidad entre las redes de los pasos de tiempo, impidiendo que de un paso a otro la solución pueda ser muy diferente.

Para medir la diferencia

Si flg_encarrilar=0 no se aplica el mecanismo explicado en el párrafo anterior.

6.20. flg_npuntosaprendidoincrementales=0

Si este parámetro se fija en 1, la cantidad de puntos a muestrear especificados en el parámetro npuntosmodeloaprendido se considera "incremental" es decir que en cada iteración se aumenta la cantidad de puntos de muestreo en esa cantidad. (ver 6.47).

6.21. flg_pcentro50=1

Si este parámetro está en 1 (valor por defecto), en las explosiones de estado (ver 5.2) para determinar en cada coordenada del estado el sentido de la variación en cada estrellita se aplica que con probabilida 50% será en el sentido creciente y con probabilidad 50% en sentido decreciente. Es decir con flg_pcentro50=1, las estrelligas se explotan manteniendo que su valor actual es el de probabilidad 50%.

Si flg_pcentro50=0, para cada coordenada del estado de cada estrillita a explotar se calcula la probilidad $p_{centro}=\frac{x_{max}-x}{x_{max}-x_{min}}$ siendo $[x_{min}, x_{max}]$ el rango de variación posible de x. Con esa probabilidad p_{centro} se decrementará el valor de x y con probabilidad $1-p_{centro}$ se drecrementará.

Si flg_pcentro50=1 entonces se impone $p_{centro}=0.5$.

Los decrementos o incrementos se calculan en base al rango posible y aplicando el concentrador_p (Ver 5.2).

6.22. flg_resample_isocf=0

Si flg_resample_isocf= 0, la explosión de las estrellitas se realiza usando el mecanismo de explosión de estados explicado en 5.2.

Si flg_resample_isocf=1, se ordenan las estrellitas del mismo paso de tiempo a explotar por orden creciente de $CF(X_i, k)$ y se muestrean estas estrellitas de forma de obtener una distribución uniforme en los valores del costo futuro y se explotan las estrellitas así muestreadas aplicando el mecanismo de explosión explicado en 5.2. Activar esta funcionalidad hace que se "presete más atención" a aquellas zonas del espacio de estado en que la variación de $CF(X_i, k)$ es mayor.

6.23. flg_resamplingbydinvar=0

Si se activa esta opción, junto con flg_simeval=1, las simulaciones de evaluación realizadas por activar flg_simeval=1 son usadas para determinar en forma dinámica cuándo es conveniente realizar una explosión de estados durante las simulaciones de exploración para recuperar la capacidad de exploración.

Si (flg_resamplingbydinvar = 1) y (flg_simeval=1), en cada paso de tiempo, para cada dimensión del espacio de estado, calcula la varianza de la posición de las estrellitas del conjunto de trayectorias de exploración (para cada semilla aleatoria) y realiza el mismo cálculo para las trayectorias de evaluación activadas por flg_simeval=1 (estas por definición debieran ser más dispersas por corresponder a semillas aleatorias diferentes). Cuando la varianza de las trayectorias exploración en alguna de las dimensiones del estado cae por debajo del 70% de la varianza de las trayectorias de simulación se dispara una explosión de estado. Independientemente de este mecanismo las explosiones de estado se rigen por los parámetros npasosm7_inf y npasosm7_sup.

6.24. flg_simeval=1

Si flg_simeval=1, luego de cada ajuste de $CF(X, k)$ en el bucle de aprendizaje, se realiza una simulación, aplciando la política de operaición aprendida $CF(X, k)$. Estas simulaciones se realizan con la información especificada en la propia Sala SimSEE, es decir que se utiliza el estado inicial del sistema especificado en la Sala (no las estrellitas iniciales de exploración) y se utiliza la semilla aleatoria inicial especificada en la sala (no el juego de semillas

usados para la exploración). Se simulan tantas crónicas (realización de los procesos estocásticos) como la cantidad NCronicas de simulación especificadas en la Sala. Para mas detalles ver 5.3.

6.25. flg_usarlassoenlugarderidge=0

En el ajuste de modelos a datos, (que es lo que se realiza durante el entrenamiento de los parámetros de la red), es común para indicar formas de penalizar el uso de los parámetros del modelo para evitar sobre-ajuste. Dos técnicas de regularización comunes son las de Ridge y la de Lasso. La primera consiste en adicionar a la función de costo la suma de los cuadrados ($J_R=\frac{1}{2}\lambda_R\sum_i \theta_i^2$) de los parámetros y la segunda en adicionar la suma de sus valores absolutos (). E $J_L=\lambda_R\sum_i |\theta_i|$ en pocas palabras, ambas penalizan el uso de los parámetros con la diferencia de que la penalización de Ridge tiende a cero dado que $\frac{\partial}{\partial \theta_i} J_R=\lambda_R\theta_i$ se anula al tender el parámetro a cero y mientras que la de Lasso tiene gradiente constante dado que $\frac{\partial}{\partial \theta_i} J_R=\lambda_R signo(\theta_i)$.

El parámetro λ_R es el fijado como pb-lr (ver. 6.51).

6.26. historydata_nrecuerdos=0

Este parámetro permite especificar que se re-utilice la información de exploración de iteraciones anteriores.

Si historydata_nrecuerdos=0 (valor por defecto) no se almacena la información de exploración de iteraciones anteriores y simplemente, se realiza el ajuste del $CF(X,k)$ en cada itearación de aprendizaje usando la nueva información de exploración. Para evitar el sobre-ajuste (overfitting) se utilzan técnicas d early-stopping y el nuevo $CF(X,k)$ austado es combinado con el anterior utilizando el parámetro learning-rate.

Si historydata_nrecuerdos=N > 0, entonces se almacena la información de exploración las N iteraciones anteriores y es utilizada junto con la nueva información para realizar el ajuste de $CF(X,k)$. El parámetro historydata_qolvido (ver 6.27) indica el peso que se le da a cada "recuerdo" respecto de la información última recibida (a la que se le da peso 1).

6.27. historydata_qolvido=0.9

Especifica el factor de olvido a aplicar sobre la información de exploración de pasos anteriores en caso de haber configurado el parámetro historydata_nrecuerdos > 0

A la información de la interación actual se le asigna peso 1. A la información de la interación anterior se la asigna peso historydata_qolvido, a la del paso anterior al anterior (historydata_qolvido)^2 y así sucesivamente.

6.28. idejecutor=12

Este parámetro es utilizado para identificar el Robot Tractorcito en un ambiente en que pueden estar varios Robot ejecutando simultaneamente sobre la misma Sala SimSEE. Las carpetas temporales que se crean para la ejecución se crean usando este identificador de forma tal que dos Robots con diferente idejecutor pueden convivir en el mismo sistema de archivo sin sobre-escribirse información entre si. En particular si se utiliza la aplicación del tractorcito con el optimizador de inversiones OddFace, se crean multiples instancias del tractorctio, según la disponibilidad de nodos de cálculo y por eso resulta necesario que cada instancia tenga un identificador.

6.29. learning-rate=0.11

En cada iteración de aprendizaje h se ajusta una nueva función $\widetilde{CF}(X,k)$ reentrenando la función $CF_{h-1}(X,k)$ obtenida en la iteración $h-1$. Para generar la nueva función $CF_h(X,k)$ se fijan sus parámetros θ_{CF_h} (pesos de las neuronas) como:

$$\theta_{CF_h} = \rho\,\theta_{\widetilde{CF}} + (1-\rho)\,\theta_{CF_{h-1}} \quad , \qquad (13)$$

Siendo $\rho =$ learnig-rate

6.30. margenearlystopping=5

Se utiliza este parámetro si se fijó manejo dinámico del earlystopping.

En caso de aplicarse el procedimiento es el que se describe a continuación.

Si la cantidad de iteraciones del algoritmo de ajuste de datos es mayor que el paso en que el ajuste del juego de datos de earlystopping ajustó mejor al modelo más el contador de earlystopping ($cnt_{EarlyStopping}$) más el margenearlystopping entonces el contador de $cnt_{EarlyStopping}$ se fija como:

$$cnt_{EarlyStopping} = max\left(margenearlystopping\,,\, cnt_{EarlyStopping} - 1\right) \quad , \quad (14)$$

Si la cantidad de iteraciones del algoritmo de ajuste de datos es mayor que el paso en que el ajuste del juego de datos de earlystopping ajustó mejor al modelo más el contador de earlystopping ($cnt_{EarlyStopping}$) más el margenearlystopping entonces el contador de $cnt_{EarlyStopping}$ se fija como:

$$cnt_{EarlyStopping} = cnt_{EarlyStopping} + 1$$

Para mas detalles vea la sección 12.

6.31. mee=11

Modo de evolución de estado. Este parámetro fija cuál de los modos de evolución de estado será el utilizado durante las simulaciones de exploración. Vea 14 para más detalles.

6.32. milo=200

Número máximo de iteraciones de aprendizaje a realizar.

6.33. modo-diff=1

Determina si se ajusta el modelo a representar la función de costo futuro o sus diferencias espaciales. Para mas detalles vea la sección 5.4.

6.34. modoearlystopping=0

Permite seleccionar el modo de control del earlystopping. Ver 12.

6.35. monitores=

No se utiliza

6.36. narmsa=1

Número de armónicos del ciclo anual a considerar en la parsimonia temporal o como entradas para las redes del tipo TANNAT.

6.37. narmsd=0

Número de armónicos del ciclo diario a considerar en la parsimonia temporal o como entradas para las redes del tipo TANNAT.

6.38. narmss=0

Número de armónicos del ciclo semanal a considerar en la parsimonia temporal o como entradas para las redes del tipo TANNAT.

6.39. ncronicas=1

Cantidad de realizaciones de los procesos estocásticos a considerar durante las simulaciones de exploración. Para cada estrellita (estado inicial) especificado, se simulan tantas trayectorias como ncronicas se especifiquen.

El formulario de la aplicación Tractorcito_GUI_rsr se permite especificar los parámetros kCronica1 y kCronica2 y la SemillaMadre. El parámetro ncroncas se calcula como:

$$\text{ncroncias} = kCronica2 - kCronica1 + 1 \quad ,(15)$$

6.40. ndimsparacontrolvardyn=-1

Este parámetro si es > 0 indica la cantidad de dimensiones a considerar para la explosión por control de varianza. Si es -1, en caso de habilitarse el control dinámico con flg_resamplingbydinvar=1, se determina las explosiones de estado por cualquiera de las variables de estado que pierda capacidad de exploración comparando su varianza con la varianza de las simulaciones de evaluación. Si ndimsparamcontrolvardyn > 0 entonces solo se controla la varianza

de las ndimsparacontrolvardyn variables de estado de mayor inercia. La inercia se intenta determinar de las simulaciones de evaluación observando la variación de las variables en el tiempo.

Este mecanismo está aún en etapa de evaluación y testeo. La idea es que las variables de estado que tienen poca inercia no son candidatas a colaborar en la formación de una política de operación y por tanto, perder la capacidad de exploración en esas dimensiones no es relevante.

6.41. nhilos=-1

Este parámetro especifica la cantidad de hilos (procesos en paralelo) se pueden utilizar para realizar las simulaciones y el entrenamiento. Con el valor nhilos=-1 se está indicando que El Tractorcito consulte al sistema operativo la cantidad de cpus (núcleos de cálculo) disponibles en el ordenador y utilice el máximo posible de hilos.

6.42. npasosfinalearlystopping=35

Indica la cantidad de pasos de earlystopping a utilizar luego de avanzado el aprendizaje.

6.43. npasosinicialearlystopping=70

Indica la cantidad de pasos de earlystopping a utilizar al inicio del aprendizaje.

$$cnt_{EarlyStopping} = n_{PFES} + (n_{PIES} - n_{PFES}) / (cnt_{iga} + 1) \quad ,(16)$$

donde $cnt_{EarlyStopping}$ es la cantidad de pasos, durante un entrenamiento que se permite que el ajuste del modelo sobre los datos seleccionados para el EarlyStopping empeore antes de cortar el entrenamiento y cnt_{iga} es el contador de iteraciones globales de aprendizaje y n_{PIES} y n_{FIES} son los valores especificados para los parámetros npasosinicialearlystopping y npasosfinalearlystopping respectivamente.

La idea es ser más permisivo en el inicio del aprendizaje en el ajuste del modelo a la nueva información y menos en la medida en que el modelo tiene una historia de ajuste.

6.44. npasosm7_inf=5

Indica la cantidad de pasos mínimos a realizar, siguiendo la dinámica del sistema, en las simulaciones de exploración en los modos en que se permite explosionar los estados antes de permitir una explosión de estados.

6.45. npasosm7_sup=90

Indica la cantidad de pasos máxima a realizar, siguiendo la dinámica del sistema, en las simulaciones de exploración en los modos en que se permite explosionar los estados antes de permitir una explosión de estados.

6.46. npots=1

Número de funciones de potencias del tiempo a considerar en la parsimonia temporal o como entradas para las redes del tipo TANNAT.

A modo de ejemplo, si $npots=3$, se considerarán las funciones temporales: t , t^2 y t^3 .

6.47. npuntosmodeloaprendido=0

Como forma de evitar el sobre-ajuste a la nueva iformación recibida luego de cada simulación de exploración, este parámetro permiteque además de la información recibida de la simulación de exploración, se realice un muestreo de la función $CF(X,k)$ aprendida en la iteración de aprendizaje anterior y que esos puntos de información sean utilizados junto con los de la simulación de exploración. La cantidad de puntos a muestrear por paso de tiempo la determina el parámetro npuntosmodeloaprendido.

Tanga en cuenta que el valor especificado como npuntosmodeloaprendido, se considera "incremental" (o sea en cada iteración de aprendizaje se incrementa la cantidad de puntos a muestrear en esa cantidad), en caso de haber fijado el parámetro flg_npuntosaprendidoincrementales=1.

6.48. ntareas=-1

Indica la cantidad de tareas a crear para las simulaciones multihilo. El valor ntareas=-1 hace que se cree la misma cantidad de tareas que de hilos.

6.49. parsimonia=3

Tipos de parsimonia temporal a aplicar. Los valores posibles son:

0: Sin Parsimonia. No se aplica ningún tipo de regularización temporal de los parámetros.

1: Parsimonia Rígida. Válida solo para el tipo de estructura TAdminEstadoCerebro. Se impone que los parámetros de las redes, de los diferentes pasos de tiempo sean un desarrollo de series de potencia y armónicos de acuerdo a lo especificado en los parámetros, npots, narmsa, narmss y narmsd.

2: Parsimonia Blanda. Se crea una función de error de ajuste de los parámetros al desarrollo de series de potencias y armónicos especificados con los parámetros, npots, narmsa, narmss y narmsd. El error de esa función es sumado a la función de error de ajuste del modelo multiplicado por el parámetro beta_global (ver 6.4)

3: Parsimonia DT. Este modo de imposición de parsimonia temporal es válido solo para la estructura TAdminEstadoCerebro. Implica crear una función de error en base a la diferencia entre los parámetros de la red de un paso de tiempo y sus adyacentes y sumar este error a la función de error usada para el ajuste de las redes multiplicaod por el parámetro beta_global (ver 6.4)

4: Parsimonia Blanda Expectation Maximization. Este modo es un experimento que no resultó y no debe utilizarse. Se deja solo por si en el futuro se quiere retomar experimentar con esta técnica. En modo muy resumido, la técnica intenta dividir el ajuste del modelo en dos etapas que se ejecutan en forma repetida. Una ajustando los parámetros con la función de error de los datos y otra ajustando los parámetros con la función de error de la parsimonia.

6.50. pb-fio=0.1;0.9;3;0.1;0.9;3 (FadeInOut)

pb_FadeInOut := prms.valVectR('pb-fio');
Ver 11

6.51. pb-lr=1E-10 (Lambda_R)

Lambda_R := prms.valFlt('pb-lr', -1);

Si Lambda_R < 0 se impone Lambda_R := prms.valFlt('pb-ah', 1.0E-12);
y como el parámetro 'pb-ah' no está implementado el valor que vale es el por defecto 1.0E-12.

Este parámetro actúa como regularizador en el uso de los parámetros de ajuste del modelo penalizando o bien la suma de los cuadrados de los parámetros o la suma de sus valores absolutos según se haya seleccionado usar regularización de Ridge o de Lasso con el parémtro flg_usarlassoenlugarderidge (ver. 6.25)

6.52. qvepp=0 (flg_QuitarVE_paso_a_paso)

Si qvepp = 1, una vez realizada una simulación de exploración, se obtienen los valores de para cada trayectoria simulada correspondiente a cada estado inicial e y para cada semilla de simulación s y en cada paso de simulación k . Cada punto de información contien:

$$\left(X\,,\,cdp\,,\,\widetilde{CF}\,,\,mee\right)_{esk} \quad , \qquad (17)$$

donde:

- X es el estado al inicio del paso,
- cdp el costo directo del paso (costo incurrido sumando todos los costos variables, fijos, penalidades),
- $\widetilde{CF}$ es la estimación de CF usada como valor del costo futuro en el estado al que evolucionaría el sistema en ese paso usada durante la simulación y
- mee es el modo de evolución de estado utilizado en ese paso de tiempo. Si el valor de mee es 7 u 11 entonces evolución se realizó en base a la dinámica del sistema y tiene sentido sumar el siguiente cdp al actual como estimación del costo incurrido en este paso y el siguiente y así sucesivamente hasta el final de la trayectoria o hasta encontrar un paso en el mee no sea ni 7 ni 11.

, y reconstruida la estimación de $\left(X_{esk}\,,CF_{esk}\right)$ que será usada para el entrenamiento del paso,

6.53. ratio_tmax_tmin=360

El parámetro ratio_tmax_tmin determina la relación entre los valores del peso de la parsimonia temporal β (ver) en caso de utilizarse la opción de parsimonia diferente por capa de neuronas que se activa o no con el parámetro flg_activarcascadebeta (ver 6.15).

```
// de los parámetros de la capa de entrada, cada neurona la hacemos Más Agil
// que la anterior
// salteo la primera que queda con beta = 1
for kNeurona := 1 to high(capa.Neuronas) do
begin
  beta := exp(-alfa * kNeurona / high(capa.Neuronas));
  kOffset := kNeurona * nParamsPorNeurona;
  for k := 0 to nParamsPorNeurona - 1 do
    beta_p[kOffset + k] := beta;
end;
```

6.54. ro_dcf=0.5

Como se explicó en 1.1, para controlar la varianza de la información relevante, se adoptaron técnicas que filtran la información de las simulaciones de exploración dejando de las mismas la correspondiente a las diferencias de la función $CF(X,k)$ entre las diferentes posibles posiciones del estado. En particular, una de las primeras transformaciones de los datos es quitarle el valore esperado a cada conjunto de información correspondiente a una misma semilla aleatoria (si marcó qvepp=1 y flg_agruparporsuerte=1) o directamente se ajusta un modelo a las diferencias y no a el valor de $CF(X,k)$ (si se marcó modo-diff=1). Por esta razón, el valor de $CF(X,k)$ no es válido como representante del valor esperado de la operación futura. El parámetro ro_dcf permite especificar el peso que se le debe dar en el ajuste a las diferencias y el complemento (1-ro_dcf) será asignado al ajuste de $CF(X,k)$ propiamente. De esa forma se logra un compromiso entre el ajuste a las diferencias (que es lo relevante a la política de operación) y el ajuste al valor esperado del costo futuro de operación, que puede resultar útil cuando se comparan resultados obtenidos por simulaciones utilizando una $CF(X,k)$, obtenida con el tractorcito que finalicen antes del horizonte final de optimización/entrenamiento. Ver 13.3.

6.55. sala=sala.ese

Especifica la Sala SimSEE objeto que se utilizará para las simulaciones de exploración y simulación.

6.56. semilla=31

Semilla madre utilizada par generar el archivo con las semillas iniciales.

6.57. sgd_maxcntiters=200

Cantidad máximas de iteraciones del algoritmo (stochastic gradient descent) usado para el ajuste de los parámetros en cada etapa de aprendizaje.

6.58. sgd_momento=0.7

Momento del algoritmo (stochastic gradient descent) usado para el ajuste de los parámetros en cada etapa de aprendizaje.

6.59. sgd_pasoini=0.01

Paso inicial del algoritmo (stochastic gradient descent) usado para el ajuste de los parámetros en cada etapa de aprendizaje.

6.60. sgd_pasominimo=0.001

Paso mínimo del algoritmo (stochastic gradient descent) usado para el ajuste de los parámetros en cada etapa de aprendizaje.

6.61. sgd-algo=adam

Tipo de algoritmo para modificación del paso usado en el algoritmo (stochastic gradient descent) usado para el ajuste de los parámetros en cada etapa de aprendizaje.

Las opciones son ADAM que implica usar el algoritmo de ADAM explicado detalladamente en la referencia [5] o RCH.

La opción RCH, es rústica y consistente en reducir el paso luego de en caso de 5 pasos consecutivos en que la función de costo crece en lugar de bajar y en aumentar el paso en el caso de 10 pasos consecutivos en que la función de costo decreció. Las reducciones del paso o el aumento se hacen dividiendo o multiplicando el paso por el conocido $\text{Golden\_Ratio} = \frac{1+\sqrt{5}}{2}$

6.62. td=-168;1

Este parámetro permite especificar las diferencias temporales a utilizar para la estimación de los valores de CF a partir de las trayectorias de exploración. Este parámetro tiene relevancia en caso de simulaciones de exploración en el modo 7, en que las trayectorias siguen la dinámica del sistema hasta el final y entonces pasa a ser relevante acumular varios pasos de costos de etapa para intentar reducir la varianza. Vea 14.3. a)

6.63. tiporn=0

Especifica el tipo de Red Neuronal a utilizar:

0: TAdminEstadoCerebro. Una Red Neuronal por paso de tiempo toda con la misma estructura. Ver 9

1: TANNAT. Una única Red Neuronal con entradas auxiliares correspondientes a las series de potencias senos y cosenos especificadas en los parámetros npots, narmsa, narmss y narmsd. Ver 10

6.64. trayectorias=trayectorias.bin

No se utiliza

6.65. xo=opt

Indica si se convoca el Tractorcito con la orden de entrar el bucle de aprendizaje (xo=opt) o solamente se desea realizar una simulación (xo=sim) sin el posterior ajuste de los parámetros a la información generada en la simulación.

7. Aplicación tractorctio_rsr

Esta es la versión de linea de comando de El Tractorctio. Debe ser llamada con el siguiente formato:

tractorcito_rsr chdir={carpeta} @=params.prm ejecutor={n}

- El parámetro chdir={carpeta}, donde {carpeta} debe indicar la ruta absoluta a la carpeta donde se encuentran los archivos de la Sala y los archivos: Semillas.xlt, EstrellasIni.xlt, CF_RN.xlt y params.prm, indica que al comienzo de la ejecución el Tractorcito cambie el directorio de trabajo a esa carpeta.
- El parámetro @=params.prm indica que se lean todos los parámetros desde ese archivo.
- El parémetro ejecutr={n}, donde {n} debe ser sustituido por un número único que identifique la instancia de ejecución del tractorcito, sirve para la ejecución en un ambiente de multiples instancias de ejecución.

8. Aspectos de la implementación en SimSEE

Como parte de la ejecución del proyecto ANII_FSE_1_2017_1_144926 "SimSEE+Variabilidad+Red+DemandaConRespuesta" / "El Tractorcito" se realizaron cambios en SimSEE para implementar el Bucle de Aprendizaje creando lo que llamamos El Tractorcito. Este nombre fue seleccionado en el sentido de "una máquina todo terreno". El bucle de aprendizaje implica una etapa de simulación de la operación del sistema en un horizonte de tiempo con una Política de Operación (PO) dada y para un conjunto de realizaciones de los procesos estocásticos, seguida de una etapa de mejora de la PO en base a la información recogida durante las simulaciones.

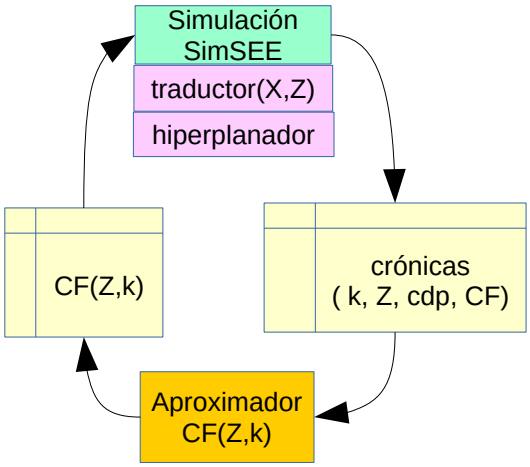


Fig. 11: Bucle de mejora de la PO.

8.1. Intermediario del Costo Futuro (ICF)

Para la implementación de El Tractorcito, se modificó el planteo de resolución del paso de tiempo original de SimSEE pasando la responsabilidad del cálculo de las derivadas $\frac{\partial}{\partial x_i} CF$ de cada Actor (generador hidráulico, banco de batería etc.) a una entidad que llamamos Intermediario de Costo Futuro (ICF) que agrega al problema de optimización una variable libre para representar el valor de CF al final del paso de tiempo y un conjunto de restricciones de desigualdad para representar la PO contenida en CF transformando en consecuencia el problema como se muestra en la ec:

$$\min_{u, X_{k+1}, CF} \left[ce(X_k, u_k, r_k, k) + CF \right] \quad \text{ec.(18)}$$
$$@ \begin{cases} u \in \Omega(X_k, r_k, k) \\ X_{k+1} = f(X_k, u_k, r_k, k) \\ CF \geq a_h^T X_{k+1} + c_h ; h = 1 \dots N_h \end{cases}$$

Donde cada par (a_h , c_h) determina un hiperplano tangente a la función $CF(X, k+1)$ siendo h un subíndice que tiene en cuenta las iteraciones de resolución del paso de tiempo. SimSEE resuelve el problema de despacho en base a una linealización del mismo, pero permite iterar el planteo de resolución del paso de tiempo para que los Actores puedan plantear mejoras al modelo lineal planteado en la iteración anterior. El ICF, aprovecha este mecanismo para ir agregando restricciones del tipo $CF \geq a_h^T X_{k+1} + c_h$ lo que permite representar el costo futuro al final del paso por una función convexa (en base a su aproximación por hiperplanos tangentes).

8.2. TAdminEstados

A los efectos de encapsular la implementación en una forma extensible en cuanto a las posibles representaciones de CF, las mismas se implementaron como Clases (Tipo de entidades) que heredan el comportamiento de una Clase Madre llamada: ***TAdminEstados***.

La clase TAdminEstados original de SimSEE representa el espacio de estado, en cada paso de tiempo mediante una discretización de cada dimensión (variable de estado) en una cantidad de puntos de discretización especificados en la definición de la Sala SimSEE. La Clase TAdminEstados tiene implementado los métodos necesarios para su utilización en el planteo del problema de la ec.18. Permite calcular la función $CF(X,k)$ y las derivadas $\frac{\partial}{\partial x_i} CF(X,k)$ como métodos virtuales lo que habilita la implementación de Clases refinadas de TAdminEstados con representación diferente de la información mientras que el refinamiento incluya la definición de esos métodos. En particular se implementaron las clases refinadas que se detallan a continuación.

9. TAdminEstadosCerebro

La Clase TAdminEstadosCerebro, representa la función $CF(X,k)$ por una red neuronal para cada paso de tiempo como se muestra en la Fig.12. La red neuronal puede tener cualquier estructura siendo sus entradas el vector de estado X y su salida el valor $CF(X,k)$. Para el cálculo del gradiente se utiliza el algoritmo de Back-propagation.

La estructura de la red, así como los tipos de regularización de sus parámetros, son definidos en forma previa al proceso de aprendizaje.

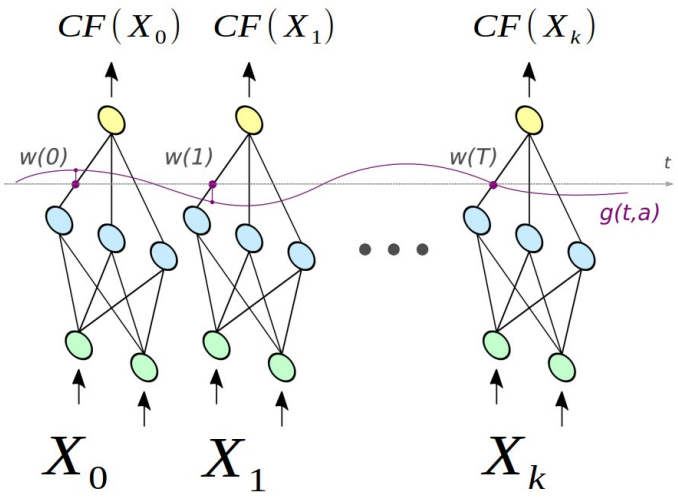


Fig. 12: Estructura TAdminEstadosCerebro

9.1. Parsimonia Temporal Rígida

Los parámetros de la RN se suponen que se explican por la suma de un desarrollo en potencias de del paso de tiempo k más un desarrollo en senos y cosenos de k :

$$\theta_k = \sum_{h=0}^{np} a_h k^h + \sum_{h=1}^{na} b_h \cos(h w_0 k) + c_h \sin(h w_0 k) \quad ,(19)$$

En la implementación realizada el usuario selecciona la cantidad de términos a utilizar tanto para la serie de potencias como para la de cosenos y senos. Como forma sencilla de introducir los parámetros, la serie de cosenos y senos se puede especificar indicando la cantidad de armónicos del ciclo anual, semanal y diario a utilizar. La idea detrás de este tipo de parsimonia, es que la PO puede cambiar progresivamente durante el horizonte de tiempo para adaptarse a cambios en la estructura del sistema o a variación prevista de los precios de los combustibles y cambiar para reflejar las estacionalidades impuestas por los ciclos anuales, semanales y diarios.

9.2. Parsimonia Temporal Blanda

El propósito es imponer un andamiento de los parámetros de la RN como el que se muestra en (19) pero en lugar de imponerlo en forma rígida, se agrega a la Función Objetivo de la optimización.

$$J=\sum_{i,k}\left(CF_{ik}-M\left(X_{ik},\theta_k\right)\right)^2+\beta_{global}\sum_k\beta(k)\left\{\theta_k-\left(\sum_{h=0}^{np}a_h k^h+\sum_{h=1}^{na}b_h\cos\left(h\,w_0 k\right)+c_h\sin\left(h\,w_0 k\right)\right)\right\}^2 \quad , \quad (20)$$

donde la primer sumatoria es la que calcula el error de aproximación del modelo (en alguna de sus variantes) y la segunda sumatoria calcula el apartamiento de los parámetros del modelo de la expresión (19). El parámetro β_{global} permite regular el peso relativo que se asigna al cumplimiento de (19) respecto del ajuste del modelo a la información de entrenamiento.

Los factores $\beta(k)$ (factores de fade-in y fade-out) permiten "ablandar" el requerimiento en forma relativa entre los pasos de tiempo. Por ejemplo utilizando valores bajos a inicio y al final de Horizonte de Optimización, se logrará dejar "mayor libertad" en los parámetros en el Horizonte de Pronóstico y en el tramo de enganche con una PO futura.

En el archivo de parámetros se puede especificar un vector "pb-fio" de seis valores. Los tres primeros regulan los pesos del FadeIn y los tres últimos los del FadeOut (inicio y fin del horizonte respectivamente).

$$pb-fio=\left[\beta_{fiA},\beta_{fiB},\alpha_{fi},\beta_{foA},\beta_{foB},\alpha_{fo}\right] \quad , (21)$$

Para ello se definen las funciones $\beta_{fi}(k)$ y $\beta_{fo}(k)$ como:

$$\beta_{fi}(k)=\beta_{fiA}+\beta_{fiB}\left(1-e^{-k_{PasosDesdeInicio}/\alpha_{fi}}\right) \quad , \quad (22)$$

$$\beta_{fo}(k)=\beta_{foA}+\beta_{foB}\left(1-e^{-k_{PasosHastaElFinal}/\alpha_{fo}}\right) \quad , (23)$$

Y la función $\beta(k)$ usada en (20) como:

$$\beta(k)=\beta_{fi}(k)\beta_{fo}(NPasos-k) \quad , \quad (24)$$

9.3. Parsimonia Temporal Diferencial

La idea es penalizar la variación en el tiempo de los parámetros, en este caso se suma a la Función de Error un término que penaliza la variación con el Paso de Tiempo k como se muestra a continuación:

$$J=\sum_{i,k}\left(CF_{ik}-M\left(X_{ik},\theta_k\right)\right)^2+\beta_{global}\sum_{j=1}^{Dim(\theta)}\sum_{k=1}^{NPasos}\beta(j,k)\left(\theta[j]_k-\theta[j]_{k-1}\right)^2 \quad , \quad (25)$$

donde la sumatoria de la izquierda realiza una estimación del error del modelo en el conjunto de observaciones y parámetros mientras que la sumatoria de la derecha penaliza la

variación en el tiempo de los parámetros. A la expresión le falta otra sumatoria correspondiente a la regularización sobre los parámetros (Ridge o Lazzo son las opciones implementadas).

El parámetro β_{global} fija el peso relativo de la penalización de la variación de los parámetros respecto de la penalización del error del modelo. Las funciones $\beta(j,k)$ permiten ponderar diferente la penalidad de variación de cada componente $\theta[j]_k$ del vector de parámetros θ_k .

Las funciones $\beta(j,k)$ permiten gran flexibilidad y para ello se implementaron formas de configuración de las mismas en modalidades simples. Para darles variación con el índice de parámetro j se previó el parámetro:

$$\text{flg\_activarcascadedbeta} = \{0 \mid 1\} \quad , \quad (26)$$

Si $\text{flg\_activarcascadedbeta} = \{0\}$, $\beta(j,k) = \beta(1,1); \forall\, j,k$,
Si $\text{flg\_activarcascadedbeta} = \{1\}$, $\beta(j,k) = \beta(j,1); \forall\, k$

Donde la función $\beta(j,1)$ diene la expresión:

$$\beta(j,1) = e^{-\alpha \frac{(j_{Neurona}-1)}{(NN-1)}}; \quad , \quad (27)$$

donde:

- $j_{Neurona}$ es el ordinal de la neurona al que pertenece el parámetro j ,
- NN es la cantidad de neuronas por paso de tiempo y
- $\alpha = \ln\left(\frac{TMáx}{TMín}\right)$. Este valor se determina fijando el parámetro "ratio_TMax_TMin" que debe representar el cociente entre la máxima constante de tiempo y la mínima de la dinámcia del sistema a capturar. Por ejemplo si hay constantes del orden anual y otras del orden diario, la relación será $\text{ratio\_TMax\_TMin} = 360$.

Mejoras futuras. a) Como se puede apreciar no se implementó variación de la penalidad por variación temporal con el paso de tiempo k . Se podría fácilmente introducir las funciones de fade In y Out aplicadas a la Parsimonia Temporal Blanda como factores multiplicadores de los $\beta(j,1)$ descriptos obteniendo así el efecto de suavizar los extremos del horizonte temporal que puede resultar beneficioso para captar efectos de pronósticos o de enganche entre POs. b) El parámetro α aprenderse a partir de la información de las trayectorias observando el decaimiento de la varianza de las variables de estado en simulaciones en el qu el mismo evoluciona con la dinámica del sistema.

10. TAdminEstadosTANNAT (Traveler Artifitial Neural Network Around Time)

La Clase TAdminEstadosTANNAT, representa la función $CF(X,k)$ por una red neuronal única para todos los pasos de tiempo. Para ello, las entradas de la red, además del vector de estado X contienen un conjunto de funciones $T(k)$ que informan a la red la posición temporal. El conjunto de funciones $T(k)$ está formado por funciones del tipo series de potencia, series de cosenos y senos (de ciclos, diarios, semanales, anuales, etc.) y de dos funciones que llamaremos FadeIn y FadeOut.

La estructura de la red y del conjunto de funciones $T(k)$ es seleccionada antes de iniciar el proceso de aprendizaje.

Las funciones del tipo serie se potencias en $T(k)$ tienen el propósito de permitirle a la PO captar cuestiones como el crecimiento vegetativo de la Demanda, o el aumento de la incertidumbre al aumentar el tiempo. Dala la experiencia sobre la información del sistema, a juicio profresional sería suficiente truncar la serie en el primer componente a lo sumo el segundo ($T(k)=\{k,k^2...\}$).

Las series de seno y coseno tiene el propósito de captar las estacionalidades. Estas estacionalidades están dadas por la Demanda y por los recursos eólico, solar, hidroeléctrico. Por la observación de los resultados, hasta un armónico 3 del ciclo anual aporta capacidad de representación y utilizar más de 5 armónicos parece excesivo.

Además del ciclo anual, en Salas de paso Diario u horario, tiene sentido intentar representar el ciclo semanal por el comportamiento de la Demanda. Nuevamente hasta el armónico 3 podría aportar información y más del 5 parece excesivo.

En salas de paso horario, la incorporación de armónicos del ciclo diario aportan información porque tanto la Demanda como los recursos eólico y solar presentan un ciclo diario marcado y nuevamente la cantidad de armónicos que recomendamos es 3 y no mucho más de 5. Por supuesto que la cantidad de armónicos a utilizar de cada ciclo es a determinar por ensayo y error, comenzando con valores bajos y agregando complejidad mientras que la PO que se obtenga sea "más óptima" que la anterior.

Por último las funciones FadeIn ($1-e^{-k/\alpha_{fi}}$)y FadeOut ($1-e^{-(N_k-k)/\alpha_{fo}}$), son del tipo de decaimiento exponencial siendo los parámetros α_{fi} y α_{fo} los que permiten regular la velocidad con que se señaliza el apartamiento del inicio o del final del horizonte de simulación respectivamente. La función FadeIn, es importante para que la red pueda tener en consideración la información de los Pronósticos (de hidrauliciadad, eólica, solar, etc.) que impactan en el origen del horizonte de simulación. La función FadeOut, es importante para permitir a la red conocer que se acerca el final del horizonte de simulación y pueda adaptar así su comportamiento para permitir un enganche suave con la PO a la que enganche (o con el final de los tiempos si no engancha con otra PO explícitamente).

El archivo de parámetros de configuración contiene un vector de seis reales 'pb-fio'. Los coeficientes 1 y 6 del vector corresponden a los parámetros α_{fi} y α_{fo} resectivamente. El resto de los coeficientes del vector no se utilizan en caso de la red TANNAT.

10.1. Series temporales.

Como ya se mencionó la idea es incluir como entradas a la red neuronal un conjunto de series temporales que puedan ser utilizadas para adaptar la red a los diferentes pasos del horizonte de simulación, captando de esa forma las tendencias y estacionalidades que correspondan así como los transitorios de partida (al inicio del horizonte) y de finalización (al final del horizonte). Con ese propósito, en la creación de la red se permite especificar la cantidad de componentes del tipo "potencia" y los armónicos anuales, semanales y diarios a considerar. También se permite especificar los parámetros para crear señales con decaimiento exponencial según la distancia al inicio y al final del horizonte de simulación.

a) Series de potencia.

La cantidad de series de potencias se especifica con un parámetro $N_{Potencias}$ en base al cual se crean las series temporales:

$$fp_h(k) = \left(\frac{(k-1)}{N_{pasos}}\right)^h ; h = 1..N_{Potencias} ; k = 1...N_{pasos} \quad \text{ec.(28)}$$

La Fig.13 muestra un ejemplo de las series temporales de potencias correspondientes a $h=1$ y $h=2$ en una simulación de horizonte semanal.

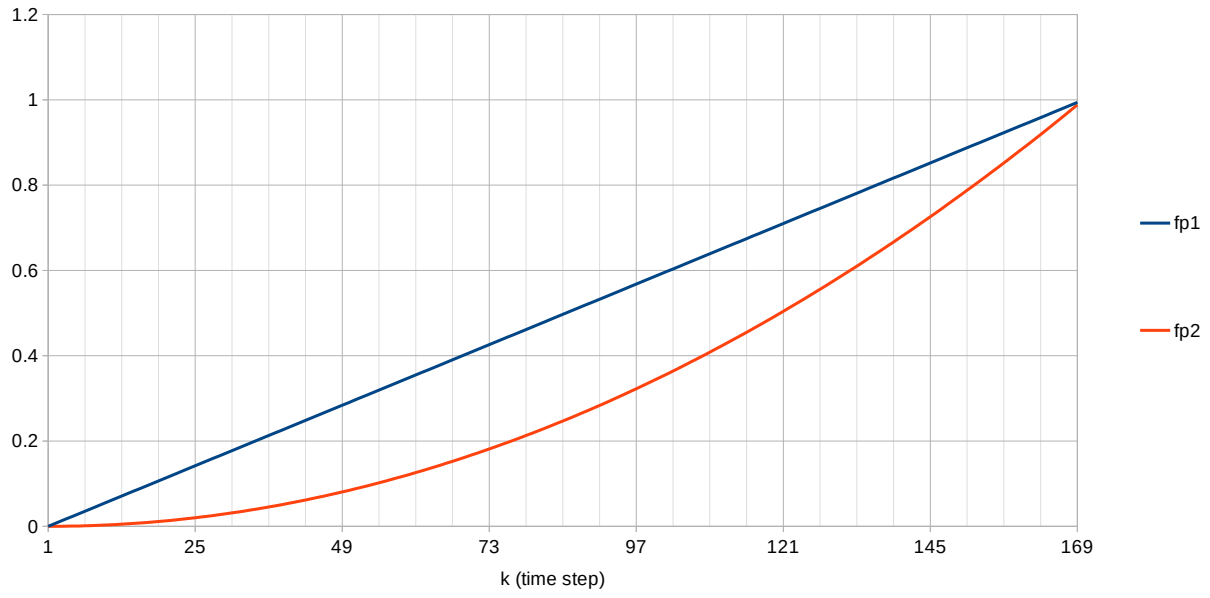


Fig. 13: Series temporales de potencia.

b) Series de armónicos.

Los armónicos se especifican indicando la cantidad de armónicos anuales, semanales y diarios.

Para cada armónico se crean dos series temporales:

$$fc_h(k)=\cos\left(2\pi h k \frac{dt_P}{dt_C}\right) \qquad \text{ec.(29)}$$

$$fs_h(k)=\sin\left(2\pi h k \frac{dt_P}{dt_C}\right) \qquad \text{ec.(30)}$$

Donde dt_P es la duración del paso de tiempo de simulación expresado días en punto flotante y $dt_C=365.2425$ para los armónicos anuales, $dt_C=7.0$ para los semanales y $dt_C=1.0$ para los diarios.

La Fig.14 muestra un ejemplo de las señales correspondientes al primer armónico semanal y diaro.

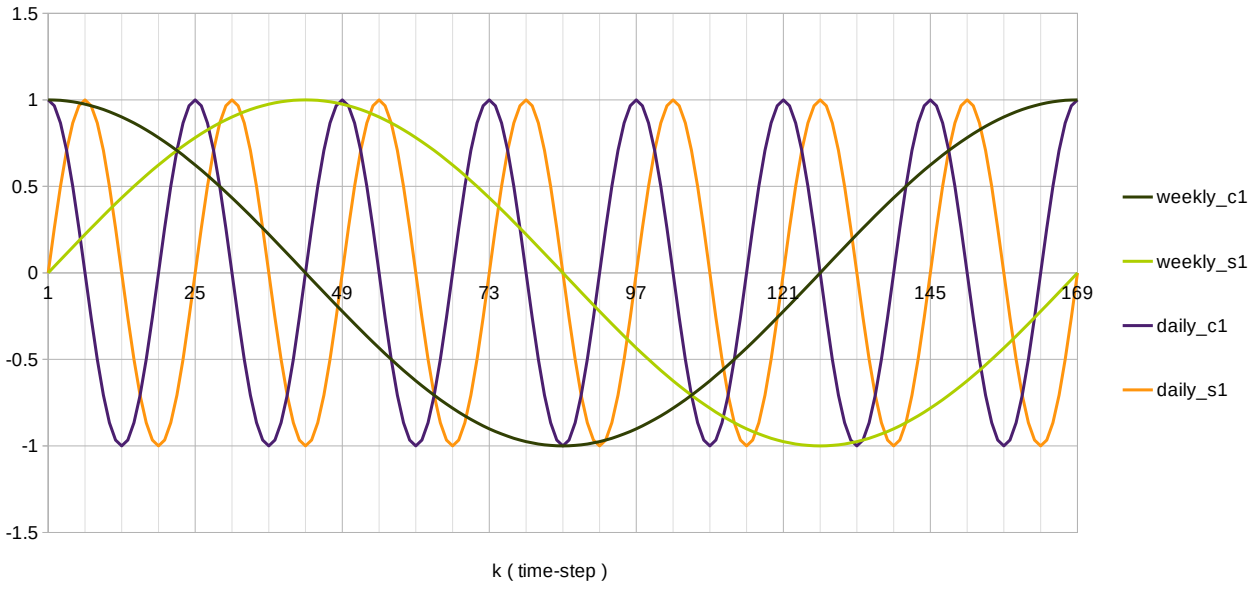


Fig. 14: Ejemplo series temporales. Primer armónico Semanal y Diario en el horizonte de 168 horas.

c) Fade In Out.

Para permitirle a TAdminEstadosTANNAT un comportamiento selectivo en el inicio del horizonte de simulación se crea la señal temporal:

$$fi(k)=\left(1-e^{-(k-1)/\alpha_{fi}}\right); \qquad \text{FadeIn ec.(31)}$$

Para permitir un comportamiento selectivo hacia el final del horizonte se crea la señal temporal:

$$fo(k)=\left(1-e^{-(N+1-k)/\alpha_{fo}}\right);$$ FadeOut ec.(32)

Dónde el paso de tiempo $k=1..N+1$ representa el tiempo (de 1 a N identifican los inicio de los pasos simulados y $k=N+1$ el final del último paso simulado.

La Fig.15 muestra las señales Fede-In y Fade-Out para un horizonte semanal y con las mismas constantes de tiempo de decaimiento $\alpha_{fi}=\alpha_{fo}=24.0$. En las simulaciones de corto plazo la señal Fade-In permite adaptar la red a la información del Pronóstico (de generación eólica/solar, etc.). En el caso de simulaciones que "empalman" con una PO al final del horizonte, si es de esperar que dicho empalme cree una distorsión (por ejemplo los espacios de estado son diferentes) entonces la señal Fade_out permite adaptar la red a dicho transitorio.

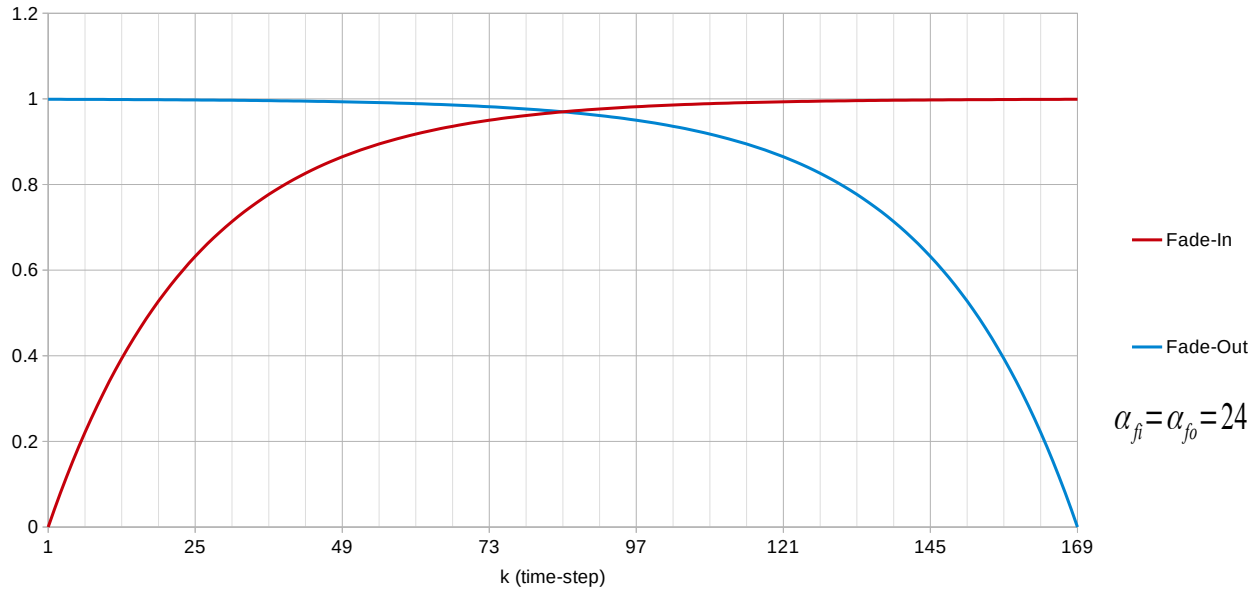


Fig. 15: Ejemplo series temporales Fade-In y Fade_Out en el horizonte de 168 horas.

TAdminEstadosKernel

La clase TAdminEstadosKernel, implementa la representación $CF(X,k)$ mediante la sumatoria de funciones del tipo Kernel. Para ello se creó la clase abstracta TKernelFuncVectRVectR que es la que utiliza TAdminEstadosKernel.

En cada paso de tiempo la representación de $CF(X,k)$ se expresa como:

$$CF(X,k)=\sum_{i=1}^{N_{Kernels}} a_i k(X,X_i)$$ ec.(33)

Donde la cantidad de kernels $N_{Kernels}$ y el tipo de kernel $k(.,.)$ determinan la capacidad de representación de la estructura y se definen como parámetros para el entrenamiento. Los centroides X_i y el peso a_i son determinados durante el entrenamiento.

A los efectos de las pruebas realizadas se definieron los siguientes tipos de Kernels (clases refinadas de TKernelFuncVectRVectR):

- TKernelGaussiano_Mat: $k(a,b)=Ce^{-\frac{1}{2}(a-b)^T M(a-b)}$, siendo M una matriz simétrica definida positiva.
- TKernelGaussiano_Diag: $k(a,b)=Ce^{-\frac{1}{2}(a-b)^T \lambda(a-b)}$ con λ una matriz diagonal (definida postiva) que regula la escala en cada dimensión del espacio.

- TKernelGaussiano_isotropica: $k(a,b)=Ce^{-\frac{1}{2\sigma^2}\|a-b\|^2}$
- TKernelConstante: $k(a,b)=C$
- TKernelPolinomico: $k(a,b)=(1+\alpha\langle a.b\rangle)^d$

La elección de un tipo de de Kernel determina el espacio de funciones que es capaz de reproducir la combinación lineal de Kernels y por tanto la capacidad de representación del modelo. A modo de ejemplo si se seleccionan Kernels polinómicos de orden 2 (como en el trabajo realizado por los colegas uruguayos [6]), nuestro modelo solo podrá representar funciones del tipo polinómicas de orden 2.

El loop de aprendizaje converge a un elemento del espacio generable pero que puede distar mucho de una PO óptima. Es claro en la aplicación a sistemas de energía eléctrica, que la función CF tiene no linealidades que llevan a que esté mal representada por funciones "muy suaves". Por ejemplo, cuando la cota de los lagos llega a niveles en que hay que abrir los vertederos (ya sea por control de inundaciones o por seguridad de la propia presa) el valor del agua embalsada (que es menos la derivada de CF respecto del volumen embalsado) es nulo y seguramente tendrá valores positivos para cotas inferiores a dicha cota de control.

La única ventaja del uso de Kernels para la representación, sería en teoría que los parámetros son determinables por una simple regresión lineal para calcular la proyección de los datos sobre la base de generación del espacio. Dado que el entrenamiento de las NN no significó un problema en las implementaciones realizadas dicha ventaja potencial no resultó relevante.

11. Atenuadores de inicio y final del horizonte temporal (Fade_IN, Fade_OUT)

Puede ser conveniente, atenuar la Parsimonia Temporal (Ver 9) al inicio y al final del horizonte de aprendizaje, liberando así la capacidad de ajuste a la información disponible al inicio y al final.

Al inicio del horizonte, por ejemplo en los primeros 10 días del despacho energético, se dispone generalmente de información de pronóstico de caudales de aportes hidráulicos, generación eólica y solar. Para un mejor aprovechamiento de esa información, liberar la capacidad de ajuste en esos 10 días, permite que la capacidad de representación del modelo dedique más recursos a ese ajuste.

Al final del horizonte de análisis, si dicho horizonte se concatena con otro de más largo plazo al cual se "engancha", entonces liberando la capacidad de ajuste arribando al momento de enganche permite que el mismo sea menos abrupto.

Con este vector de valores se especifica la relajación de las regularizaciones temporales al inicio y al final del paso de tiempo, en el caso de estar aplicando parsimonias temporales en base a

regularización o simplemente se crean las señales temporales para señalizar el inicio y el final del horizonte de optimización para el caso de las redes tipo TANNAT.

Los tres primeros valores corresponden al inicio del horizonte y los últimos 3 al final, e indican el factor atenuante que señaliza el inicio y el fin siendo, de cada grupo de tres valores definen una evolución exponencial desde el primer valor al segundo valor en la cantidad de pasos de tiempo indicadas por el último valor.

Los valores del vector representan entonces el siguiente juego de parámetros:

$$\beta_{fiA}, \beta_{fiB}, \alpha_{fi}, \beta_{foA}, \beta_{foB}, \alpha_{fo} \quad , \qquad (34)$$

Si $k = 1,\ 2,\ \ldots N_{Pasos}$ son los pasos de simulación, el atenuador de la regularización efectivo se calcula como:

$$\beta_{efectivo}(k) = \beta_{fi}(k)\,\beta_{fo}(k) \quad , \qquad (35)$$

siendo

$$\beta_{fi}(k) = \beta_{fiA} + \beta_{fiB}\left(1 - e^{-(k-1)/\alpha_{fi}}\right) \quad , \qquad (36)$$

el atenuador de entrada y

$$\beta_{fo}(k) = \beta_{foA} + \beta_{foB}\left(1 - e^{-(N_{Pasos}-k)/\alpha_{fo}}\right) \quad , (37)$$

el atenuador de salida.

Si se quiere desactivar alguno de los atenuadores, se debe poner el α correspondiente en -1 (menos uno).

12. Early Stopping (ES)

El entrenamiento de un modelo en base a un conjunto de observaciones se corre el riesgo de que se produzca lo que se conoce como sobre-ajuste (overfitting). El sobre-ajuste del modelo a los datos de entrenamiento lleva a la pérdida de generalización del modelo. Esto es, frente a un nuevo conjunto de datos, el modelo presenta un error mayor que el que se obtendría si no se hubiese producido el sobre-ajuste a los datos de entrenamiento. Este es un problema clásico, cuanto más parámetros tiene un modelo (mayor capacidad de representación) mayor es la posibilidad de que se produzca sobre-ajuste.

Es clásico dividir la información disponible, en un conjunto de entrenamiento y otro de verificación y realizar la calibración del modelo usando el conjunto de entrenamiento pero chequeando con el de verificación. El concepto de Early Stopping es precisamente ese. Seleccionar un grupo de de control y avanzar en el entrenamiento "vigilando" el comportamiento sobre el grupo de control.

En [7] se presenta el concepto de Early Stopping y tres medidas de "cuando parar". Estas técnicas están planteadas para el entrenamiento de un modelo en base a un conjunto de datos. En la referencia [8] se aplica en Early Stopping en un loop de aprendizaje que podría considerarse similar al del aprendizaje de una PO. En esta referencia se concluye que el Early Stopping mejora la eficiencia del aprendizaje, lo que podemos confirmar empíricamente que también sucede en nuestro caso.

La referencia [8] presenta el uso de Early Stopping sobre un problema de aprendizaje de un modelo para la predicción del pico de Demanda diaria. Tiene la particularidad de que la función de costo, usada para el ajuste de los parámetros, está relacionada con maximizar la información explicada por el modelo (en base a maximizar la entropía) en lugar de la minimización el error cuadrático medio. El criterio de detención Early Stopping usado es buscar el mínimo de la medida del error sobre el conjunto de control.

En método de Early Stopping consiste en seleccionar al inicio de la etapa de entrenamiento (en cada iteración del bucle de aprendizaje) un subconjunto de las trayectorias resultantes de la simulación que serán usadas como grupo de control. Las trayectorias no seleccionadas para control serán las utilizadas para el grupo de entrenamiento. El parámetro "fearlystopping" determina el factor del conjunto de trayectorias que serán seleccionadas para formar el grupo de control. Por ejemplo fearlystopping = 0.1 significa que el 10% de las trayectorias simuladas se utilizarán como grupo de control.

Nuestra aplicación tiene dos particularidades a resultar: 1) La información recibida en cada etapa de exploración/simulación depende de la aproximación anterior a la función CF y 2) las exploración del espacio de estados realizada en cada paso por el conjunto de trayectorias de cada etapa es "pobre" en el sentido de que realizar una exploración exhaustiva significa caer en la Maldición de la Dimensionalidad de Bellman. Esto tiene consecuencias sobre el aprendizaje. Ambos aspectos alimentan la idea de NO sobre ajustar a los datos de entrenamiento. Porque los mismos son imperfectos (estimaciones de CF en base a la última simulación) y poco densos en cuanto a representar CF(X).

Los criterios de parada por Early Stopping implementados se basan en usar la misma función de error usada para el entrenamiento de los parámetros del modelo para medir el error sobre el grupo de control. Se inicia el algoritmo de entrenamiento con un ContadorDeEarlyStopping en un valor dado y cada vez que el error sobre el grupo de control resulta superior al del paso anterior (en los pasos del algoritmo Stochastic Gradient usado para el entrenamiento) se decrementa en 1 dicho contador. Cuando el contador llega a cero se detiene el entrenamiento.

Como parámetros al mecanismo de EarlyStopping implementado, se pueden especificar los parámetros:

- fearlystopping=0.1 Factor del conjunto de trayectorias que formarán el grupo de control.
- npasosinicialearlystopping=70 Valor inicial a utilizar como ContadorDeEarlyStopping en la primer iteración del bucle de aprendizaje.
- npasosfinalearlystopping=35 Valor al que tiende el valor a utilizar como ContadorDeEarlyStopping a medida que avanzan las iteraciones del bucle de aprendizaje de acuerdo a la fórmula:

$$, \quad cnt_{ES} = n_{PasosFinalES} + \left(n_{PasosInicialES} - n_{PasosFinalES} \right) / NIters_{Aprendizaje} \qquad (38)$$

- modoearlystopping=0 Este parámetro determina el modo de criterio de EarlyStopping, como ya se mencionó, cada vez que la medida del error sobre el grupo de control crece respecto al paso anterior (del algoritmo de la etapa de entrenamiento) se decrementa el ContadorDeEarlyStopping y cuando llega a cero se detiene la etapa de entrenamiento,

continuando con el bucle de aprendizaje.
Si **modoearlystopping=0**, simplemente se decrementa en 1 el valor cnt_{ES} y cuando llega a cero se detiene el entrenamiento y se retorna el último valor evaluado del juego de parámetros.
Si **modoearlystopping=1** se va registrando el error sobre el grupo de control y guardando el mínimo valor y el juego de parámetros correspondiente. Cada vez que el error sobre el grupo de control es superior al del paso anterior se decrementa en 1 cnt_{ES} y si el error es menor que el mínimo registrado, el contador cnt_{ES} se retorna al valor inicial (comienza nuevamente la cuenta regrsiva). El entrenamiento termina cuando $cnt_{ES}=0$ (o se supera el número máximo de iteraciones o el paso se reduce por debajo del umbral fijado). Como juego de parámetros se devuelve aquel que durante la etapa de entrenamiento resultó en una menor medida del error sobre el grupo de control. Este último criterio corresponde al utilizado las referencias antes citadas y tiene la lógica de devolver el juego de parámetros que mejor "generaliza" dado que minimiza el error sobre el grupo de control. En nuestro caso, empíricamente (midiendo el costo esperado sobre simulaciones con las PO obtenidas con ambos criterios).
Si **modoearlystopping=2** se aplica el mismo criterio que con modeearlystopping=1 pero en forma adicional se modifica el valor cnt_{ES} o el número máximo de iteraciones $Max_{CntIters}$ para la siguiente etapa de entrenamiento. Si el fin del entrenamiento se produce por haber alcanzado la cantidad máxima de iteraciones (y no por $cnt_{ES}=0$) entonces se incrementa en 100 el valor $Max_{CntIters}$ si el fin ser produce por $cnt_{ES}=0$ entonces si $cnt_{Iter}>k_{MenorErrorControl}+2\,\widetilde{cnt}_{ES}$ entonces para la próxima iteración se considera $\widetilde{cnt}_{ES}=max(5,\widetilde{cnt}_{ES}-1)$. Esto es se decrementa el umbral de cuenta de aumento del error sobre el grupo de control si la distancia entre la cantidad de iteraciones y la iteración de menor error sobre el grupo de control es mayor al doble del umbral $\widetilde{cnt}_{ES}$ vigente. Por el contrario, si $cnt_{Iter}\leq k_{MenorErrorControl}+5$ entonces se incrementa el umbral $\widetilde{cnt}_{ES}$ en una unidad.

En nuestro caso, se implementaron dos modalidades de Early Stopping:

1. Early Stopping Cuenta Regresiva Mejor Control (ESCRMC).
2. Early Stopping Cuenta Regresiva Ultima Evaluación (ESCRUE)

13. Bucle de aprendizaje.

Por Bucle de Aprendizaje entendemos la ejecución repetida de las siguientes etapas:

1. **Simulación/Exploración**. A usando una función de costo futuro dada $CF_i(X,k)$ se simula la operación del sistema para un conjunto de estados iniciales y un conjunto de realizaciones de los procesos estocásticos. Como se describe en la sección
2. **Mejora de CF**. Con la información recolectada durante la etapa de simulación, se mejora la representación del coto futuro, ajustando la estimación obteniendo así una nueva función $CF_{i+1}(X,k)$.

Dada una función $CF_0(X,k)$ cualquiera, es posible simular trayectorias de evolución del estado resolviendo en forma repetida la ec.39 para una secuencia de las entradas no controlables $R_k=\{r_k, r_{k+1}, ...\}$ dada. Observar que para la resolución del paso k se supone conocido solo el elemento r_k de las entradas no controladas.

$$\min_{u, X_{k+1}} \{ce(X_k,u_k,r_k,k)+CF(X_{k+1},k)\} \quad @\begin{vmatrix} u\in\Omega(X_k,r_k,k) \\ X_{k+1}=f(X_k,u_k,r_k,k)\end{vmatrix} \qquad \text{ec.(39)}$$

Realizada la simulación para un conjunto de estados iniciales $\{X_0^p\}$ y de secuencias $\{R_0^h\}$ se simula para todas las combinaciones de estado inicial $p=1,2,...N_X$ y secuencia de entradas $h=1,2,...N_R$, simulando un total de N_X*N_R crónicas (o realizaciones, o futuros posibles).

La información útil para la Política de Operación se encuentra en las derivadas direccionales $\frac{\partial}{\partial X}CF(X,k)$ por lo cual más que el valor de la función $CF(X,k)$ nos interesa representar las diferencias espaciales $CF(X_a,k+1)-CF(X_b,k+1)$ para poder comparar el costo de $ce(X,r,u,k)$ con la afectación del CF por movimiento del estado en la resolución del problema de optimización planteado en la ec.17 para el paso k .

En la Fig.6 se presenta, en un ejemplo real, la dispersión que tiene el Costo Futuro de un estado dado. Observar que al muestrear esa distribución, para un mismo estado de llegada al final del paso k se pueden obtener muestras equiprobables con diferencias importantes seguramente superiores a lo que puede ser el costo de una etapa $ce(X,r,u,k)$. Este es un problema clásico de estimación de la diferencia entre dos variables aleatorias por método de Montecarlo y para reducir la cantidad de muestreos necesarios se recurre a una técnica conocida como Common Randon Numbers (CRN) [9].

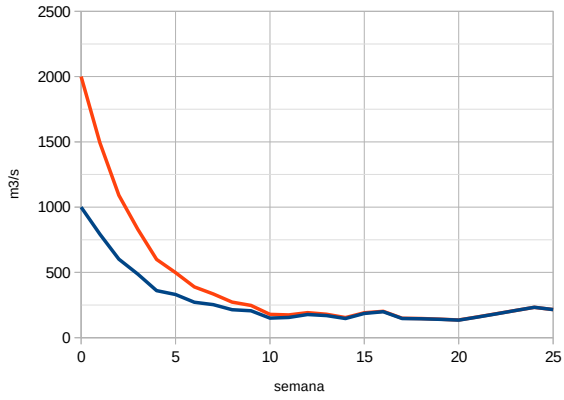


Fig. 16: Convergencia de las trayectorias.

La plataforma SimSEE está diseñada para modelar los procesos estocásticos de forma de poder garantizar un muestreo con CRN en las diferentes trayectorias usando para ello la técnica estándar de inicializar los modelos estocásticos en base a una *semilla aleatoria* que determina la

generación del generadores de números pseudo-aleatorios usados en la base de los modelos estocásticos en la implementación del software.

Durante las simulaciones, inicializando el sistema en diferentes estados $\{X_i\}$ y los modelos de los procesos estocásticos con la misma semilla aleatoria, se logra simular y evaluar el costo futuro desde estados diferentes para una misma realización de los proceso estocásticos y tener estimaciones así de $\langle CF(X_i,k)\rangle_h$ para los diferentes estados de partida $\{X_i\}$ en el conjunto de semillas aleatorias simuladas enumeradas con el subíndice h .

Dado que los sistemas suelen ser estables, partiendo de dos estados iniciales diferentes, al simular, con el pasar del tiempo, con la misma realización de los proceso estocásticos es altamente probable que las trayectorias del sistema converja y terminen juntándose. Esta convergencia será diferente según las variables de estado. Aquellas variables asociadas a constantes de tiempo mayores, demorarán más pasos de tiempo en aglutinarse mientras que las asociadas a constantes de tiempo menores lo harán en menos pasos. A partir de que las trayectorias se aglutinan, como el sistema estará sometido exactamente a las mismas entradas se convierten en una sola trayectoria. A modo de ejemplo, la Fig.16 muestra lo que podrían ser las trayectorias del caudal afluente, medio semanal, a la hidroeléctrica Bonete de Uruguay. Como se puede apreciar, partiendo dos estados bien diferentes, y sometido el sistema a las mismas lluvias (realización de los procesos estocásticos), al cabo de 10 semanas las trayectorias no son diferenciables. Esta convergencia de las trayectorias sometidas a las mismas entradas no controlables (las cuales incluyen los procesos estocásticos) tiene como consecuencia, que de seguir la dinámica del sistema en la evolución de los estados, las simulaciones van perdiendo capacidad de exploración del espacio de estado.

Los movimientos del estado en un paso de tiempo están limitados por la dinámica del sistema y a menor paso de tiempo menor será el movimiento posible del estado durante el paso de tiempo. Es así que a menor paso de tiempo menor será el valor del costo de etapa $ce(X,r,u,k)$ (pues integra los gastos de un paso de tiempo de menor duración) y menor la diferencia de los estados alcanzables según las decisiones (por ser menor la variación del estado). Por tanto, a menor paso de tiempo, más comprometida será la detección correcta de las diferencias $CF(X_a)-CF(X_b)$ a comparar con $ce(X,r,u,k)$, dado que la varianza de $CF(X)$ para un estado dado es independiente del paso de tiempo utilizado. Dada la relación entre la varianza de CF en los sistemas de generación de energía eléctrica y los valores de $ce(X,r,u,k)$ (como se muestra en las Figs. 6 y 9 para el caso del sistema uruguayo) las simulaciones que no usen técnicas de reducción de varianza como la CRN están condenadas a fracasar (no logran convergencia a una Política de Operación).

Para intentar tener una PO óptima, es necesario explorar CF(X) en todas las regiones del espacio de estado. En este sentido, aunque se parta de estados diferentes, la convergencia de las trayectorias al aplicar el mismo conjunto de realizaciones de los procesos estocásticos (condición directamente asociada a la necesidad de usar CRN) lleva a que siguiendo la dinámica del sistema se pierda la capacidad de exploración. A modo de ejemplo, si se parte de 100 estados iniciales y 5 semillas aleatorias se simularán 500 trayectorias que en grupos de 100 irán convergiendo a 5 trayectorias. A su vez, la técnica CRN solo se puede aplicar sobre los 5 grupos de trayectorias con igual realización de los procesos estocásticos.

La pérdida de la capacidad de exploración de las trayectorias que comparten la misma realización de los procesos estocásticos, obliga a limitar el horizonte de tiempo durante el cual se evoluciona el estado de acuerdo a la ecuación de evolución de estado ec.4. En la implementación realizada de la etapa de simulación, se permite especificar cuantos pasos seguidos (parámetro TD de las siglas en inglés Time Difference) se utiliza la ec.4 para evolucionar el estado. Es durante esos pasos que es posible en la etapa de Mejora de CF estimar el costo futuro como el costo acumulado de los costos de etapa más el valor $CF_i(X,k)$ para el estado al final de la última etapa del grupo con la función $CF_i(X,k)$ utilizada para la simulación:

$$CF_{i+1}(X_k,k)=\sum_{l=0}^{TD-1} q^l ce(X_{k+l},u_{k+l},r_{k+l})+q^{TD}CF_i(X_{k+TD},k+TD) \qquad \text{ec.(40)}$$

Luego de $TD-1$ pasos con evolución usando la dinámica del sistema, en la implementación realizada se realiza una "explosión de los estados" logrando así separarlos nuevamente para volver a recuperar la capacidad de exploración.

En un extremo, con $TD=1$, al final de cada etapa se realiza una explosión de los estados manteniendo así constante la capacidad de exploración. Esto parecería óptimo en el sentido de asegurar una buena exploración del espacio de estado, pero tiene el inconveniente, de enlentecer la convergencia a la PO óptima por imponer que la información de las consecuencias sobre el futuro, viajen hacia el presente al ritmo de un paso de tiempo por iteración. La ec.40, determina cuantos pasos, desde el futuro hacia el presente, viaja la información en la etapa de Mejora de CF. Si se utiliza $TD=1$, la información viajará solo un paso de tiempo durante cada etapa de Mejora de CF. Esto implica, por ejemplo que si se está simulando con paso diario tres años de la operación del sistema, se necesitarán por lo menos 1095 iteraciones del Bucle de Aprendizaje para que la Política de Operación de los primeros pasos de tiempo haya recibido por lo menos una vez la información de las consecuencias de llegar al final del horizonte con el estado en determinada región del espacio de estado.

El horizonte de tiempo que importa para razonar sobre la velocidad de propagación de la información está directamente relacionado con las constantes de tiempo de la dinámica del sistema.

Si la dinámica del sistema es tal que las acciones del presente puedan tener efecto en los costos dentro de 3 años (por ej. capacidad de embalse pluri-anual) entonces, será necesario que la información de por lo menos 3 años alcance a las primeras etapas de la simulación.

Se llegan entonces a un compromiso en el que por una parte es necesario inicializar trayectorias desde estados diferentes para lograr explorar diferentes regiones, pero las trayectorias deben alimentarse con entradas usando CRN para que los valores obtenidos sirvan en la estimación de las diferencias de CF y al hacer esto, las trayectorias se irán concentrando hacia una única trayectoria.

Lo anterior llevó a diseñar el proceso de simulación con varios modos de evolución del estado en cada paso de tiempo que se detalla en la sección 5.2.

13.1. Aproximación a la función de Valor con nivel libre

Si el objetivo es aproximar un modelo $y=CF(X,\theta)+\alpha$ ajustando el conjunto de parámetros θ y el parámetro α , problema se podría plantear como:

$$\min_{\theta,\alpha} J_v(\theta,\alpha)$$ ec.(41) Aproximación a la función de Valor.

Dónde: $J_v(\theta,\alpha)=\sum_i e_i^2$, $e_i=CF(X_i,\theta)+\alpha-y_i$

En el óptimo se debe cumplir:

$$\frac{\partial J_v(\theta,\alpha)}{\partial\theta}=\sum_i 2\,e_i\frac{\partial e_i}{\partial\theta}=0$$ ec.(42) .

y

$$\frac{\partial J_v(\theta,\alpha)}{\partial\alpha}=\sum_i 2\,e_i\frac{\partial e_i}{\partial\alpha}=\sum_i 2\,e_i=0$$ ec.(43) .

Las condiciones de optimalidad sobre la ec.43 implican:

$$\sum_i e_i=0$$ ec.(44)

y sobre la ec.42 implican:

$$\sum_i e_i\frac{\partial e_i}{\partial\theta}=\sum_i (CF(X_i,\theta)+\alpha-y_i)\frac{\partial e_i}{\partial\theta}=\sum_i (CF(X_i,\theta)-y_i)\frac{\partial e_i}{\partial\theta}+\sum_i\frac{\partial e_i}{\partial\theta}\alpha$$ ec.(45)

La ec.44 implica $\sum_i\frac{\partial e_i}{\partial\theta}=0$

De la definición del error e_i se puede escribir $\frac{\partial e_i}{\partial\theta}=\frac{\partial(CF(X_i,\theta)-y_i)}{\partial\theta}$

Y sustituyendo ambas expresiones en la ec.45 se obtiene

$$\sum_i e_i\frac{\partial e_i}{\partial\theta}=\sum_i (CF(X_i,\theta)-y_i)\frac{\partial(CF(X_i,\theta)-y_i)}{\partial\theta}=0$$ ec.(46)

Observar que la condición de optimalidad expresada por la ec.46 es independiente de α por lo que se puede resolver el problema 41 para un valor de α fijo y luego calcular α de forma tal de cumplir con la ec.44 como se muestra en la ec.

$$\alpha=\frac{1}{N}\left(\sum_i^N y_i-\sum_i^N CF(X_i,\theta)\right)$$ ec.(47)

Para que los parámetros θ no sean desaprovechados en tratar de explicar lo que luego puede explicar el parámetro α se debería cumplir que

$$\frac{\partial \alpha}{\partial \theta}=\frac{1}{N}\sum_{i}^{N}\frac{\partial CF(X_i,\theta)}{\partial \theta}=0$$ ec.(48)

Ejemplo 1d.

Sistema real $y=12+x^2$

Muestras (x, y) : (0, 12), (2, 16).

Aproximación por un hiperplano y = a x + c

e1 := 12 - c, e2:= 16 - (2a + c)

grad_a := (16 -(2a+c)) 2 = 0

grad_c :=(12-c) + (16 - (2a+c)) =0

sol: c = 12 , a = 2

y = 2 x + 12

****

13.2. Aproximación a las diferencias espaciales desde cualquiera de los estados visitados.

La información de la Política de Operación, se encuentra en las diferencias de la función CF entre diferentes estados. Teniendo en cuenta lo anterior, planteamos el problema de ajuste de los parámetros de la representación como un problema de optimización con una función objetivo formada por el error en el estimación de las diferencias de CF entre los diferentes estados visitados como se muestra a continuación:

$$\min_{\theta} J_d(\theta)$$ ec.(49) Aproximación a las variaciones espaciales de CF.

Dónde: $J_d(\theta)=\frac{1}{4N^2}\sum_{ij}\left((CF(X_j,\theta)-CF(X_i,\theta))-(y_j-y_i)\right)^2$

que puede escribirse en función del error de la función de valor como:

$$J_d(\theta)=\frac{1}{4N^2}\sum_{ij}(e_i-e_j)^2=\frac{1}{4N^2}\sum_{ij}(e_i^2-2e_ie_j+e_j^2)=\frac{1}{4N^2}(\sum_{ij}e_i^2-\sum_{ij}2e_ie_j+\sum_{ij}e_j^2)$$

$$J_d(\theta)=\frac{1}{4N^2}(\sum_{ij}e_i^2-\sum_{ij}2e_ie_j+\sum_{ij}e_j^2)=\frac{1}{4N^2}(N\sum_{i}e_i^2-2(\sum_{i}e_i)(\sum_{j}e_j)+N\sum_{j}e_j^2)$$

$$J_d(\theta)=\frac{1}{4N^2}\sum_{ij}(e_i-e_j)^2=\frac{1}{2N}\sum_i e_i^2-\frac{1}{2N^2}\left(\sum_i e_i\right)^2=J_v(\theta)-\frac{1}{2N^2}\left(\sum_i e_i\right)^2$$

lo que se simplifica en:

$$J_d(\theta)=J_v(\theta)-\frac{1}{2}\left(\frac{\sum_i e_i}{N}\right)^2 \qquad \text{ec.(50)}$$

El gradiente de $J_d(\theta)$ resulta entonces:

$$\frac{\partial J_d(\theta)}{\partial \theta}=\frac{\partial J_v(\theta)}{\partial \theta}-\frac{1}{N^2}\left(\sum_i e_i\right)\left(\sum_i \frac{\partial e_i}{\partial \theta}\right) \qquad \text{ec.(51)}$$

Observar que la condición de optimalidad del problema de aproximación a la función de valor implica $\frac{\partial J_v(\theta)}{\partial \theta}=0$ y por tanto $\frac{\partial J_d(\theta)}{\partial \theta}=-\frac{1}{N^2}\left(\sum_i e_i\right)\left(\sum_i \frac{\partial e_i}{\partial \theta}\right)$ y no necesariamente es nulo. Por lo tanto los problemas no son equivalentes.

Observar también, que si el error promedio es nulo, $\sum_i e_i=0$ entonces se cumple $J_d(\theta)=J_v(\theta)$

Considerando el problema de la ec.49, el cálculo del gradiente $\frac{\partial J_d(\theta)}{\partial \theta}$ se puede realizar usando el algoritmo BackPropagation. Para cada muestra presentada a la red, es posible calcular el error e_i y extraer su derivada $\frac{\partial e_i}{\partial \theta}$ imponiendo un 1.0 (uno) como error en la salida del algoritmo BackPropagation e ir calculando la suma ponderada de los gradientes por los errores para el cálculo de $\frac{\partial J_v(\theta)}{\partial \theta}$ y la suma simple del error e_i y de los gradientes $\frac{\partial e_i}{\partial \theta}$ para poder calcular la ec.51 una vez presentado el conjunto de muestras a considerar.

¿Porqué esto importa?. Para entender la importancia de usar la ec.49 en lugar de la ec.41 como objetivo del entrenamiento hay que pensar que estamos entrenando con información MUY ESCASA y que al ser así, la importancia que le damos al valor absoluto de cada nuevo conjunto de muestras introduce una distorsión. Solo a modo de ejemplo, si la nueva información es solo un nuevo punto (resultado de simular solo una crónica), en la ec.41 tendrá un efecto en el entrenamiento mientras que en la ec.49 se cumple la condición de optimalidad (por estar calculada sobre una única muestra es trivial que el lado izquierdo de la ec.51 es nulo y es el mínimo valor que puede tomar una suma de cuadrados) y por tanto no se producen cambios en los parámetros de la RN.

13.3. Combinación de objetivos (ro_dCF)

Lo importante a la política de operación es la información contenida en las derivadas direccionales de la función de costo futuro como ya se explico. Igualmente, resulta deseable que el valor de la función represente una estimación del costo futuro esperado. Para ello, nos planteamos tomar como función objetivo una combinación de las ecuaciones 41 y 49. Para ello agregamos un parámetro que indica el peso ρ_{dCF} que damos a la ec.49. La función objetivo sería entonces:

$$J(\theta)=\rho_{dCF} J_d(\theta)+(1-\rho_{dCF}) J_v(\theta) \quad , \text{(52)}$$

y utilizando las ec.50 y 51 podemos escribir:

$$J(\theta)=J_v(\theta)-\frac{1}{2}\left(\frac{\sum_i e_i}{N}\right)^2 \rho_{dCF} \qquad \text{ec.(53)}$$

y su gradiente:

$$\frac{\partial J(\theta)}{\partial \theta}=\frac{\partial J_v(\theta)}{\partial \theta}-\frac{1}{N^2}\left(\sum_i e_i\right)\left(\sum_i \frac{\partial e_i}{\partial \theta}\right)\rho_{dCF} \qquad \text{ec.(54)}$$

13.4. Aproximación a las diferencias espaciales desde cualquiera de los estados visitados por Grupos de Muestras.

Las simulaciones se realizan en base a conjunto de crónicas que pueden estar agrupadas en crónicas de igual semilla aleatoria para la realización de los procesos estocásticos durante la simulación. Las crónicas de cada grupo se diferencian solamente por el estado inicial del sistema.

Para reducir la varianza de las estimaciones de las diferencias $d_{ij}=CF(X_i,k)-CF(X_j,k)$ solo se consideran en la función de error las diferencias entre estados del mismo Grupo de Crónicas y la fórmula del error y su gradiente quedarían escritas como:

$$J_d(\theta)=\frac{1}{N_g}\sum_{g=1}^{N_g}\left(J_{vg}(\theta)-\frac{1}{2N^2}\left(\sum_{i=1}^{i=N} e_{ig}\right)^2\right) \qquad \text{ec.(55)}$$

El gradiente de $J_d(\theta)$ resulta entonces:

$$\frac{\partial J_d(\theta)}{\partial \theta}=\frac{1}{N_g}\sum_{g=1}^{N_g}\left(\frac{\partial J_{vg}(\theta)}{\partial \theta}-\frac{1}{N^2}\left(\sum_{i=1}^{i=N} e_{ig}\right)\left(\sum_{i=1}^{i=N}\frac{\partial e_{ig}}{\partial \theta}\right)\right) \qquad \text{ec.(56)}$$

Para el caso de querer entrenar una única red para todos los pasos de tiempo, los Grupos de Muestras tienen que ser por paso de tiempo y si corresponde, en cada paso de tiempo, las muestras a su vez tiene que agruparse por la agrupación de crónicas creadas por la semilla aleatoria.

14. Modos de Simulación/Exploración

Dada una función $CF(X,k)$ es posible simular la evolución del sistema desde un estado inicial dado generando resolviendo el problema de despacho (3) secuencialmente para los sucesivos pasos de tiempo generando así así lo que llamamos una trayectorias.

O está en las derivadas espaciales $\frac{\partial}{\partial x_j} CF(X,k)$ y por consiguiente, la capacidad de capturar información a los efectos de la PO de un conjunto de trayectorias, está en que las mismas "exploren" diferentes regiones del espacio de estado. Si dos trayectorias, no se diferencias para una dimensión x_j del estado, las mismas no aportan información relevante para estimar la derivada parcial $\frac{\partial}{\partial x_j} CF(X,k)$.

Por razones de control de la varianza de las estimaciones de $CF(X,k)$, en las simulaciones utilizaremos la técnica de Common Randon Numbers (CRN) [9]. Esto lleva a que partiendo de dos estados iniciales, la resolución del problema de despacho (3), sometido a la misma realización de los procesos estocásticos lleve a que las trayectorias converjan a una sola (perdiendo la capacidad de exploración).

Para la definición de las trayectorias a simular, se selecciona un conjunto de estados iniciales (que llamaremos Estrellitas) y un conjunto de semillas aleatorias iniciales (que llamaremos simplemente Semillas). Las Semillas, inicializan los generadores de números aleatorios del simulador y eso lleva a que fijada un asemilla que de determinada la realización de las entradas no controlables $\left\{r_0, r_1, \ldots r_k, r_{k+1}, \ldots\right\}$.

```
type
 TModoEvolucionEstado = (
   CMEE0_Dinamica,   // Dinámica del sistema; X = Xs


   CMEE1_Cilindro,   // No Evoluciona; X = X Cilindro.


   CMEE2_ExplosionEnSim_LOCA,   // Explosión_LOCA
fExplosionEstado.Reiniciar(globs.semilla_explosion_estados * 100031 + globs.kTrayectoria);

          //       al explotar en base al kTrayectoria parecería no respetar el tema de REDUCCION
DE VARIANZA
```

// el CMEE4 parece estar mejor en ese esentido.

// La explosión se produce al evolucionar estado durante la simulación.

CMEE3_Cilindro_MOVIL, // No evoluciona; X = Cilindro. En Cada iteración se modifica el cilindro sorteando un nuevo conjunto

// de estrellitas iniciales

CMEE4_Explosion_X0_EnIter_y_X_EnSim_CRN, // Exploción_CRN;

// Explota las estrellas iniciales X0, en cada iteración y luego la X durante la simulación

// fExplosionEstado.Reiniciar(globs.semilla_explosion_estados * 100031 + globs.jEstadoInicial);

// La explosión se produce al evolucionar estado durante la simulación.

CMEE5_Rotativo3240, // Repite el ciclo CMEE3, CMEE2, CMEE4, CMEE0

CMEE6_ExplosionEnTrayectoria_submodo, // X <- X próximo paso desde la trayectoria salvo el útlimo paso que hace X = Xs por no poder leer un punto más

CMEE7_Dinamica_submodo, // Dinámica del sistema; X = Xs. Evoluciona igual que el CMEE0_x, nada más que este se utiliza como un sub-modo del CMEEB_x

CMEE8_Rnd67, // cntIter_local < 3) or (fuenteM3.rnd < 0.3) Resamplea las trayectoras como el CMEE6 e impone modo = CMEE8

// en caso contrario resamplea como el CMEE7 e impone modo = CMEE9

CMEE9_Rnd67, // Dinámcia X = Xs, Aplica los mismo que el CMEE8

CMEEA_M6ActualizadoSGD, // En el resampleo funciona con el CMEE6. (OJO-ME PARECE QUE FALTA ALGO)

// En este modo se le define un procedimiento de CALL_BACK al entrenador SGD de forma

// que lo llama y van cambiando los datos de entrenamiento.

// Parece una forma de tratar de realimentar nuesvas estimaciones

// de CF directamente sobre las trayectorias durante el proceso de entrenamiento. Debe haber sido una forma de intentar

// hacer que la información fluya hacia el presente en la exploración en CMEE6 pero si ese fue el intento

// falta cargar en el CF de las trayectorias la nueva estimación.

CMEEB_MixDinExplosion, // Tramos con dinámica y explosiones a cada tanto. Durante la simulación lee de la trayectoria el tipo de

// evolución a considerar para el punto.

// Según el sub_modo escrito en las trayectorias, evolucióna con el CMEE6_x (si ese es el sub-modo) o

// evoluciona con la dinámica del sistema en cualquier otro caso.

CMEEC_no_usado,

CMEED_no_usado,

CMEEE_no_usado,

CMEEF_NO_DEFINIDO // Se usa solo para indicar que no se ha asignado un valor específico

);

14.1. Programación dinámica aproximada

En la lucha contra la Maldición de Bellman han surgido diferentes formas de obtener aproximaciones a una política óptima de operación de un sistema dinámico. Los libros [10] y [11] son buenos ejemplos del arsenal disponible.

14.2. Agregaciones

Con este término se conoce a las diferentes estrategias para reducir la dimensión de los espacios a explorar. Las estrategias van desde las más clásicas como son la agregación en escalas temporales encadenadas a reducción del espacio de estado o del espacio de las variables no controlables mediante transformaciones que en base al conocimiento experto representen super-estados del sistema equivalente.

Estas estrategias se aplican en forma independiente de si el algoritmo de búsqueda de una política óptima de operación se realiza con la Recursión de Bellman o por otros métodos.

a) Agregaciones espaciales

Por agregaciones espaciales nos referimos a la simplificación del espacio de estados del sistema a los efectos del cálculo de la función $CF_{\check{P}O}(Z,k)$ en lugar de calcular $CF_{\check{P}O}(X,k)$ siendo el estado "comprimido" Z de menor dimensión que X y la transformación $Z=M_R(X)$ la que define la reducción de estado. A modo de ejemplo, en Brasil, en el modelo de

Largo Plazo, se agrupan las centrales hidroléctricas por región y se calculan doce "embalses equivalentes" y se crea así la política de operación correspondiente a un vector de estado de dimensión 13 (los 12 embalses más la condición hidrológica del Niño/Niña). En la jerga del área de aprendizaje por refuerzo, este tipo de agregación sería asimilable a la representación del estado en un espacio de características.

Si solo se realizan simulaciones forward (hacia adelante en el tiempo) se puede representar en la simulación el estado completo X y utilizar la transformación de reducción en la resolución del problema de despacho que quedaría:

$$u = \underset{u}{argmin}\{ce(X,r,u,k) + q\,CF_{\tilde{PO}}(M_R(f(X,r,u,k)),k+1)\} \quad \text{ec.(57)}$$

En el ejemplo de Brasil, utilizan el software Newave que utiliza la téncia SDDP (Stochastic dual dynamic programming) y por tanto las simulaciones son siempre hacia adelante.

Si se quisiera utilizar la Recursión de Bellman, el algoritmo es necesario lograr una discretización $I_Z(k)$ del estado comprimido Z y representar la posición real del sistema como:

$$X = M_A(Z,w) \quad \text{ec.(58)}$$

Siendo $M_A(Z,w)$ una transformación de ampliación (o amplificación) del estado comprimido que transforma un estado comprimido dado en el conjunto de estados posibles $X/Z = M_R(X)$ siendo w una entrada que permite identificar cada punto X de dicho conjunto. En el caso de SimSEE donde se aplica este tipo de reductores, se asegura que la probabilidad condicional P(X/Z) se reproduce si el conjunto se muestrea siendo w ruido blanco con distribución $N(0,1)$. Considerando esta nueva fuente de aleatoriedad, la recursión de Bellman para determinar la función de Costo Futuro sobre el estado comprimido estaría representa por el siguiente algoritmo:

FOR $k = k_{último}$ **DOWNTO** $k = k_{primero}$ **DO** ec.(59)
  FOR $Z_i \in I_Z(k)$ **DO**
    BEGIN
      $a := 0;$
      FOR $r \in R_k$ **DO**
      BEGIN
        $X = M_A(Z, w(r));$
        $a := a + \underset{u}{min}\{ce(X,r,u,k) + q\,CF_{\tilde{PO}}(M_R(f(X,r,u,k)),k+1)\};$
      END
      $CF_{\tilde{PO}}(X,k) := \frac{a}{n(R_k)};$
    END

Donde la función $w(r)$ genera realizaciones con distribución $N(0,1)$ a partir de las realizaciones de r .

Reducción del estado de los procesos estocásticos

La reducción más común aplicada durante la optimización es la de la disponibilidad de los equipamientos (generadores, líneas de transmisión, etc.). En la mayoría de los aplicaciones de uso industrial, para la determinación de la política de operación se suele realizar sorteos de

disponibilidad en forma independiente del estado de cada elemento. Esto es una reducción que mapea el sub-espacio del estado del sistema asociado a los estados de disponibilidad en un espacio de dimensión nula. En este caso, la transformación de ampliación $M_A(Z,w)$ no depende de Z y corresponde simplemente a la realización de sorteos independientes para reflejar el estado disponible/indisponible con las probabilidades de estado estacionario. En la medida en que los elementos del sistema sean muchos, cada uno de ellos en forma individual resulta marginal y por tanto esta simplificación suele aceptarse como adecuada en sistemas grandes. En simulaciones de corto plazo con detalle horario, la información de disponibilidad, se incorpora en forma determinística indicando qué unidades están disponibles y cuales no y en el caso de unidades indisponibles cuándo se prevé vuelvan a estar disponilbes.

Otro ejemplo común de reducción total del estado es el relativo a los procesos estocásticos que representan la generación eólico o solar. Durante la optimización se suelen representar estos procesos por su valor esperado según la hora del día y la transformación $M_A(Z,w)$ no depende de Z y corresponde a generar la distribución de probabilidades del recurso en cada hora del día.

De los procesos estocásticos representados, el más relevante para los sistemas eléctricos de Latinoamérica es el que representa los caudales afluentes a las hidroeléctricas por la alta componente de este tipo de generación en sus países y porque los regímenes de lluvia presentan estancionalidades anuales y volatilidad de largo plazo presentando años secos o años lluviosos. De acuerdo al conocimiento sobre el clima, la temperatura superficial del océano Pacífico en la región conocida como N34 impone un sesgo en las precipitaciones con una permanencia inter-anaual.

En el caso de Uruguay, el proceso estocástico que representa los caudales y el iN34 (ver [12]) está modelado por modelo del tipo CEGH (ver [13]). Este tipo de modelos permite definir transformaciones de reducción del estado del proceso (en un espacio transformado gaussiano donde el proceso es representado por un sistema lineal recursivo). Estas reducciones corresponden a agregaciones espaciales como las mencionadas. La facilidad de los modelos CEGHs es que permiten definir las transformaciones M_R y M_A como transformaciones lineales.

Información de pronósticos en los procesos estocásticos

La información de pronósticos (de caudales, generación eólica, solar, temperatura, etc.) forma parte del estado del sistema en cuanto a que es información que determina su futuro sesgando el comportamiento de los proceso estocásticos modelados. Pero al igual que los cronogramas de incorporación/des-afectación de unidades de generación o las tendencias de crecimiento de la Demanda, esta la información de pronóstico puede considerarse "escrita en el tiempo", en lugar de aumentar la dimensión del estado del sistema.

A modo de ejemplo, en las implementaciones que utilizan modelos de Markov para representar posibles realizaciones de los caudales a las represas, el conocimiento del pronóstico se puede considerar iniciando los estados según la condición real del sistema y cambiando en forma acorde las probabilidades de transición entre los estados de acuerdo a la información del pronóstico. En los modelos con estados continuos como los CEGHs usados en SimSEE, la incorporación de los pronósticos tiene un tratamiento similar, en base al cálculo de una serie de sesgos atenuadores como se explica en [13].

b) Agregación en escalas temporales encadenadas

Es común aplicar directamente la Recursión de Bellman diferenciando en por lo menos tres escalas temporales Largo Plazo (LP), Mediano Plazo (MP) y Corto plazo (CP). Para ello se construye un modelo del sistema para cada horizonte. A modo de ejemplo, la Tabla 1 muestra la

cadena de modelos actualmente en uso en ADME para la programación del despacho energético de Uruguay en las diferentes escalas temporales.

Tabla 1: Ejemplo de agregación temporal (Uruguay).

Modelo	***Horizonte***	***Paso de tiempo***	***Variables de estado***
LP	20 años	semanal	V_{Bon}, i_{N34}
MP	1.5 años	diario	$V_{Bon}, V_{Pal}, V_{SG}, h_{RN}, h_{RU}, i_{N34}$
CP	10 días	horario	$V_{Bon}, V_{Pal}, V_{SG}, h_{RN}, h_{RU} i_{N34}$

Donde:

- V_{Bon} , V_{Pal} y V_{SG} son los volúmenes embalsados en las hidroeléctricas Bonete, Palmar y Salto Grande respectivamente,
- h_{RN} y h_{RU} representan el estado de escurrimiento de las cuencas del Río Negro y del Río Uruguay respecitvamente e
- i_{N34} es el índice que representa la anomalía de la temperatura superficial del océano Pacífico en la región N34.

El Modelo LP se ejecuta por lo menos dos veces por año (Programación Estacional) y se obtiene del mismo la función: $CF_{LP}(V_{Bon}, i_{N34}, k)$

Esta función de Costo Futuro permite la simulación con paso semanal del modelo LP y adicionalmente se utiliza para inicializar la función $CF_{MP}(V_{Bon}, V_{Pal}, V_{SG}, h_{RN}, h_{RU}, i_{N34}, k_{semana})$ al final del último paso de tiempo del horizonte del modelo MP. La inicialización se realiza mameando los volumenes embalsados en el modelo de MP en volumen de Bonete que representa la misma energía almacenada. La variable i_{N34} se mapea en forma directa y las variables h_{RN} y h_{RU} que están presentes en el modelo MP y no en el LP se mapean de forma tal que $\frac{\partial}{\partial h_{RN}} CF_{MP} = \frac{\partial}{\partial h_{RU}} CF_{MP} = 0$ reflejando así que no se dispone información respecto de las derivadas de CF en esas direcciones. De esta forma se logra que la información de las derivadas direccionales de CF_{LP} se transmitan en la inicialización de CF_{MP} .

Resuelta la etapa de optimización del modelo MP, se obtiene la función de costo futuro $CF_{MP}(V_{Bon}, V_{Pal}, V_{SG}, h_{RN}, h_{RU}, i_{N34}, k_{día})$ que es utilizada para las simulaciones en el horizonte MP para programación del uso de los recursos, proyección de evolución de las cotas de los lagos, programación de las importaciones de combustibles, etc. Adicionalmente dicha función es utilizada para inicializar los valores de función de la función $CF_{MP}(V_{Bon}, V_{Pal}, V_{SG}, h_{RN}, h_{RU}, k_{hora})$ al final del último paso de tiempo del horizonte de optimización para comenzar la recursión de Bellman. Como se puede apreciar, las variables de estado del modelo CP son las mismas que las del MP, salvo i_{N34} dado que en el horizonte CP (diez días), la temperatura superficial del océano se puede considerar constante.

14.3. Programación Dinámica Aproximada

Con la denominación Programación Dinámica Aproximada existen un abanico de soluciones. Una primer clasificación posible es entre los métodos que se basan en la aproximación de la función de valor del estado o función de Coso Futuro $CF(X,k)$ o la función de valor-acción $Q(X,u,k)$ que representa el valor presente esperado de la operación futura para un estado dado y para una acción (vector de control) dado en un paso de tiempo dado. Las soluciones basadas en la función de valor-acción son las adecuadas cuando no se conocen las ecuaciones del modelo del sistema y el Agente (que va aprendiendo una política de operación en la terminología de aprendizaje automático) debe aprender por ensayo y error tanto la política como el modelo. En el caso de los sistemas de energía eléctrica las ecuaciones de los modelos son conocidas y la dimensión del vector de control u es ordenes de magnitud superior a la dimensión del estado del sistema. Por esta razón en este trabajo solo se consideran los métodos basados en la aproximación de la función de valor.

Las soluciones planteadas en base a aproximar $CF(X,k)$ implican una propuesta inicial $CF_0(X,k)$, la realización de simulaciones hacia adelante resolviendo el despacho con la ec.18 y recolectando los costos incurridos para mejorar la de $CF(X,k)$ obteniendo así $CF_1(X,k)$ para la próxima iteración. Las soluciones difieren en la forma en que se proponen las simulaciones que determinan las trayectorias exploradas del estado, la forma en que la información recolectada se utiliza para mejorar la propuesta de $CF(X,k)$ y en la forma en que se realiza la aproximación de $CF(X,k)$.

a) Formas de exploración y actualización de CF

Las técnicas de aproximación de $CF(X,k)$ en base partir de una propuesta inicial $CF_0(X,k)$, pueden describirse como un bucle de iteraciones formado por las etapas:

- **Exploración.** En la que se simulan un conjunto de Trayectorias del sistema en base a la resolución del despacho inducida por la aproximación $CF_j(X,k)$ disponible en la iteración j. Durante la simulación se recolecta la información de los estados visitados, el costo de etapa y la estimación $CF_j(\widetilde{X},k)$ en el estado de llegada $\widetilde{X}$ al final de cada paso de tiempo en que se resolvió el despacho.
- **Actualización.** En esta etapa se estiman los valores del costo futuro, al inicio de cada paso de tiempo en cada trayectoria teniendo en cuenta los costos de etapa y las estimaciones del costo futuro en los estados de llegada resultantes de las simulaciones.
- **Aproximación.** Con la información actualizada en la etapa anterior, se dispone de un conjunto de estados visitados para cada paso de tiempo y de una estimación del costo futuro de ese estado. Con ese conjunto de puntos de información se re-entrena el modelo que representa la función de costo futuro ajustando el conjunto de parámetros del modelo.

Dinámica natural del sistema

La exploración más trivial es la que produce la propia dinámica del sistema ec.4 sometiendo al sistema desde un estado inicial X_0 a diferentes realizaciones de las entradas no controlables

$\{r_0, r_1, ...r_k, ...\}_h$. Esta solución tiene la virtud de reflejar las probabilidades de visitar los estados para la política de operación dada, pero presenta por lo menos dos inconvenientes:

1. podría no explorar nunca porciones del espacio de estado que no fueron alcanzadas en el proceso de iteraciones de mejora de la política de operación y
2. la varianza de $CF(X,k)$ asociada a las diferentes realizaciones de $\{r_0, r_1, ...r_k, ...\}_h$ es ordenes de magnitud superior al valor del costo de etapa $ce(X,r,u,k)$ lo que atenta contra la resolución del problema de despacho ec.18.

Actualización tipo TD(n) (Time Differences)

Usando la ecuación de evolución del sistema permite utilizar los costos de etapa de cada trayectoria para calcular la nueva estimación del costo futuro en cada estado visitado, simplemente sumando los costos de etapa durante N_{TD} pasos de tiempo y sumando el valor de la estimación $CF_j(X_{k+N_{TD}}, k+N_{TD})$ correspondiente al estado de llegada al final de dichos pasos de tiempo ponderado por el correspondiente factor de actualización $q^{N_{TD}}$. Este procedimiento de actualización es lo que se denomina $TD(N_{TD})$ de las siglas en inglés de Time Differences como se muestra en la siguiente ecuación:

$$CF_{j+1}(X_k^h, k) = \sum_{p=0}^{N_{TD}-1} q^p ce(X_{k+p}^h, r_{k+p}^h, u_{k+p}^h, k+p) + q^{N_{TD}} CF_j(X_{k+N_{TD}}^h, k+N_{TD}) \quad \text{ec.(60)}$$

Donde el superíndice h identifica la trayectoria (que a su vez corresponde a una realización concreta de las entradas no controlables) y el subíndice j representa la iteración del bucle de aprendizaje.

Como primer observación sobre la ec.60 se observa que la información contenida en $CF_j(.,k+N_{TD})$ viaja en el tiempo N_{TD} pasos del futuro hacia el presente y participa de la nueva propuesta $CF_{j+1}(.,k)$. Entonces, a mayor valor de N_{TD} más rápido serán visualizados en el presente las condiciones del futuro.

Como segunda observación, a mayor valor de N_{TD} mayor será la varianza de la suma de los costos de etapa que son exclusivos de la realización.

Actualización TD_compuesta

En el libro [14] se le llama simulación de Montecarlo al caso en que la actualización $TD(N_{TD})$ se aplica en cada paso hasta el final del horizonte simulado y se menciona el problema de la varianza introducida por los costos de etapa acumulados. Valores de N_{TD} bajos implican que la información viaje del futuro al presente en más iteraciones del bucle de aprendizaje pero que la varianza de la suma de costos de etapa impuesta por las crónicas simuladas tenga menos influencia. Esto lleva a plantear una solución que combina varias actualizaciones del tipo $\sum_i p_i TD(N_{TDi})$ con diferentes pesos.

Trayectorias sin evolución natural

En lugar de partir de un estado inicial dado, la idea es partir de un conjunto de estados $I_X^j(k)$ seleccionados al azar en cada iteración de aprendizaje para cada paso de tiempo. Con este

criterio solamente se puede aplicar actualización $TD(1)$, dado que el sistema no sigue su evolución de acuerdo con la ecuación de evolución de estados (ec.4), sino que en cada paso de tiempo se parte de una posición del estado pre-asignada y se resuelve el despacho obteniedo así el costo de la etapa y el estado al que evolucionaría el sistema en caso de seguir la ec.4 lo que permite calcular el valor $CF_j(X_{k+1}, k+1)$ con la aproximación del costo futuro utilizada en la simulación j . Una ventaja de este método de exploración es que con el pasar de las iteraciones se habrá explorado todo el espacio de estado.

Un caso particular de la exploración sin evolución natural sería el caso en que $I_X^j(k)=I_X^j(0)$ en el que el conjunto de estados iniciales de todos los pasos se considera el mismo. A este caso se le podría llamar "trayectorias cilíndricas". El mantener los estados constantes puede ser útil para intentar entender el aprendizaje observando los valores de las trayectorias, pero respecto al caso de estados variables presenta menos riqueza de exploración, dado la información coleccionada en un paso de tiempo se propaga hacia el presente (un paso de tiempo) al realizar la actualización de la estimación del costo futuro.

Modo mixto

En este modo, las trayectorias se originan a partir de un conjunto de estados iniciales $I_X^j(0)$ seleccionados al azar y se permite la evolución natural del sistema (ec.4) durante N_{en} pasos de tiempo. Para el paso $N_{en}+1$ se construye un nuevo conjunto $I_X^j(N_{en}+1)$. Este método une las ventajas de la evolución natural en cuanto a permitir aplicar $TD(N_{TD})$ hasta un valor $N_{TD}=N_{en}$, permitiendo así que la información del futuro viaje hacia el presente a un ritmo superior a un paso por iteración y permite "explotar" el conjunto de estados de las trayectorias a cada N_{TD} , separando los estados y manteniendo así la capacidad de exploración.

La necesidad de utilizar este modo mixto se percibirá mejor en la sección en que se trata el uso Common Random Numbers como técnica de reducción de varianza y cómo dicha técnica implica que con la evolución natural las trayectorias convergen (es decir los estados de las diferentes trayectorias se van aproximando) perdiendo por tanto capacidad de exploración.

Stochastic Dual Dynamic Programming (SDDP)

La técnica de SDDP es la utilizada principalmente en Brasil. En esta sección se realiza una descripción informal del método intentando dar los conceptos que permitan captar su funcionamiento. En la referencia [15] se presenta el método en forma detallada y aplicación al despacho de sistemas hidro-térmicos. El método consiste en observar que el problema de despacho ec.18 puede relajarse considerando una función de costo futuro que minore a la realmente buscada. De esta forma comenzando un $J(X,k)=cte<CF_{PO}(X,k)$ el problema:

$$CF_J=\min_u\{ce(X,r,u,k)+qJ(f(X,r,u,k),k+1)\} \qquad \text{ec.(61)}$$

Es una relajación del problema original y por tanto la solución obtenida $\check{CF}_j$ es una cota inferior del valor del costo futuro $CF_{PO}(X,k)$ que se obtendría si se resolviera el problema

orignal. Adicionalmente, al resolver el problema relajado (ec.61) para todos los pasos de tiempo se obtienen los respectivos vectores de control u_k y costos de etapa $ce(X_k, r_k, u_k, k)$. Realizando la suma actualizada de los costos de etapa desde el futuro hasta el paso k se obtiene una evaluación $\hat{CF}_j$ del costo futuro correspondiente a la ejecución de la operación (esto es una evaluación en la función de costo futuro verdadera de la solución obtenida con el problema relajado).

Los multiplicadores de Lagrange correspondientes a $X = X_k$ permiten calcular la derivada $\frac{\partial}{\partial X}\check{CF}_j$ en la solución relajada y portando (bajo el supuesto que las funciones de costo futuro son convexas) es posible "desrelajar" (esto es ajustar la relajación) del problema resuelto agregando el hiperplano que pasa por $X = X_k, CF = \check{CF}_j$ como una restricción de desiguladad que va definiendo el supremo del problema.

En cada iteración se obtiene una nueva cota inferior $\check{CF}_j$ y superior $\hat{CF}_j$ y un nuevo plano tangente. Si la diferencia entre la menor cota superior y mayor cota inferior es suficientemente pequeña se considera que el problema ha convergido y se tiene una función de Costo Futuro (dada por el conjunto de puntos del conjunto de hiperplanos que no es superado por ninguno de los hiperplanos en la dimensión CF). De esta forma en base a iteraciones en las que se va ajustando la relajación se va aproximando la función de costo futuro obteniendo cota inferior y superior en los estados visitados. En particular, en el estado de partida las cotas de $CF_{PO}(X_0, 0)$ pueden utilizarse para terminar las iteraciones. Si la función de costo no es convexa, las cotas superior e inferior no convergen a un único valor sino que tienen a una diferencia constante que se denomina "gap de dualidad".

El método es elegante en cuanto al formalismo matemático que permite el planteo en base a una iteración de relajaciones. En la práctica lo anterior es cierto cuando los procesos estocásticos no juegan un papel importante en el sistema. El manejo de la variabilidad se realiza en base a definir un modelo de estados de Markov, con probabilidades de transición entre los estados, modelo que genera un árbol de escenarios a partir del instante inicial según las combinaciones posibles de los eventos relevantes. En la práctica, este árbol de escenarios no se puede dejar crecer considerando todas las combinaciones en cada paso de tiempo pues lleva a una explosión combinatoria de los estados a considerar rápidamente. Las iteraciones de relajaciones sucesivas se debe realizar sobre los nodos del árbol de escenarios. En la práctica esto lleva a que la representación de lo estocástico sea simplificada para hacer el problema tratable.

Rolling Horizonts

Esta técnica se menciona en el presente trabajo solo para dar un panorama de las soluciones existentes, pero no está dentro de las soluciones propuestas dado que no incluye la representación de una función de Costo Futuro.

La técnica Rolling Horizonts consiste en explorar a partir de un estado dado X_k y de un conjunto de realizaciones posibles de las variables no controlables $\left\{r_k^h, r_{k+1}^h, \ldots r_{k+n_{RH}-1}^h\right\}$, las trayectorias posibles (identificadas por cada realización h) del sistema siguiendo su evolución natural durante un horizonte temporal pre-establecido de n_{RH} que incluye varios pasos de tiempo consecutivos. En base a esa exploración se selecciona el vector de control u_k que guía al sistema por la trayectoria de menor costo y se permite la evolución del sistema un paso de tiempo y se vuelve a repetir el proceso. Un desarrollo detallado de la técnica puede verse en [16]. Esta técnica parece eficiente en la medida en que el sistema tenga constantes de tiempo inferiores o iguales a las capacidades de pronóstico de las variables de interés. Esto permite realmente resolver la

exploración minimizando los costos conociendo las realizaciones en el horizonte. Algunos trabajos (ver [17]) plantean soluciones para aplicar la técnica en sistemas con constante de tiempo diferentes.

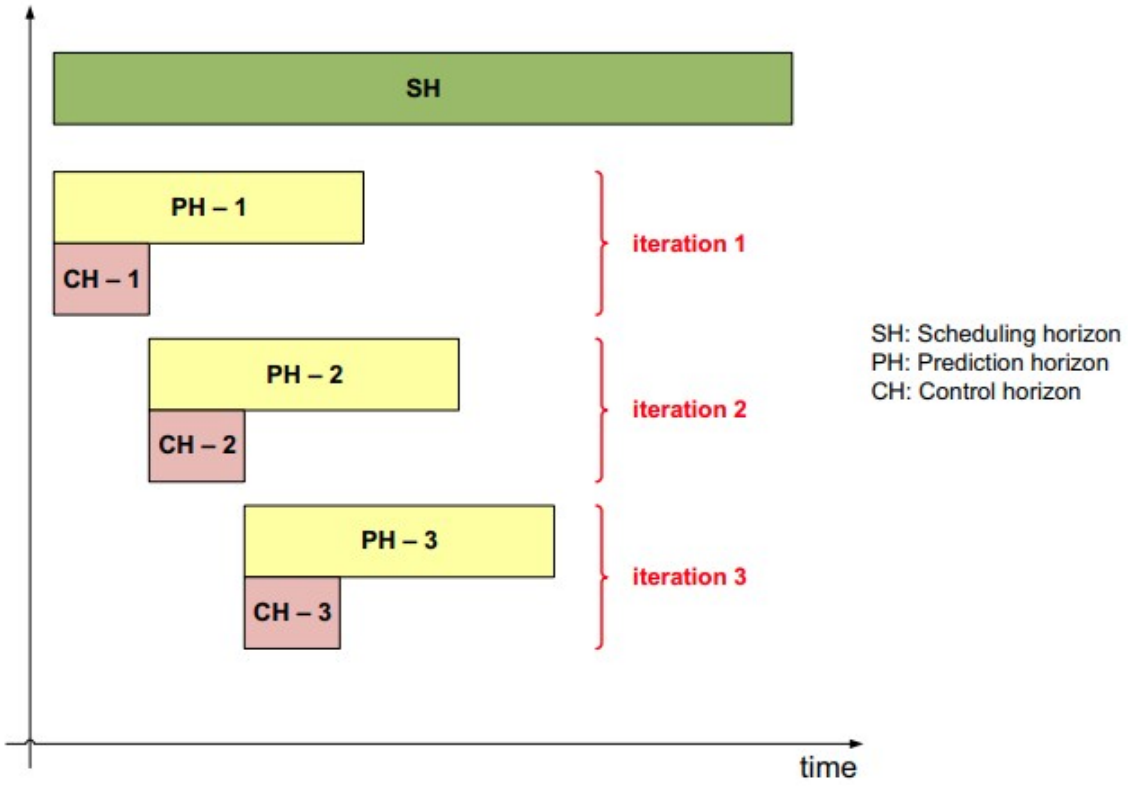


Fig. 17: Marco de trabajo para técnica Rolling-Horizont. Fuente: [18]

15. Anexo. Desarrollo Mixture Density Networks

La verosimilitud de las muestras se puede expresar como se muestra en la ec.62

$$L=\prod_k p(y_k, X_k)=\prod_k p(y_k/X_k)\,p(X_k) \qquad \text{ec.(62).}$$

Suponemos que la función de densidad de probabilidad condicional $p(y/X)$ puede expresarse por la combinación lineal convexa de la ec.63 dónde las funciones $\alpha_i(X)$ tienen la expresión de la ec.67, siendo los valores $\gamma_i(X)$ las salidas de una NN al igual que los parámetros $\beta_i(X)$ que determinan las funciones $\phi_i(y_k/X_k;\beta_i(X_k))$.

$$p(y_k/X_k)=\sum_i \alpha_i(x_k)\phi_i(y_k/x_k;\beta_i(x_k)) \qquad \text{ec.(63).}$$

Se trata entonces de entrenar la NN para determinar los $\gamma_i(X)$ y $\beta_i(X)$ que maximizan la verosimilitud (ec.62) o lo que es lo mismos, minimizan menos el logaritmo de la verosimilitud como se expresa en la ec.64

$$-\ln(L)=-\sum_k \ln(p(y_k,X_k))=-\sum_k \ln(p(y_k/X_k))-\sum_k \ln(p(X_k)) \quad \text{ec.(64).}$$

El término $\sum_k \ln(p(X_k))$ de la ec.64 es independiente de los parámetros de entrenamiento, por lo cuál no participa del problema de minimización el que se puede plantear como se muestra en al ec.65.

$$min(\sum_k E_k) \quad \text{ec.(65).}$$

Dónde E_k se calcula como se muestra en la ec.66.

$$E_k=-\ln(\sum_i \alpha_i(X_k)\phi_i(X_k)) \quad \text{ec.(66).}$$

$$\alpha_i(X)=\frac{e^{\gamma_i(X)}}{\sum_j e^{\gamma_j(X)}} \quad \text{ec.(67).}$$

Para cada muestra presentada a la NN durante el entrenamiento, es posible calcular el aporte a la función de costo E_k como se expresó en la ec.66 y sus derivadas respecto a las salidas de la RN para ingresar dichos valores como “errores” en el algoritmo de BP (back propagation) para calcular el gradiente de la función objetivo respecto de los parámetros de la NN.

$$B(x_k)=\sum_i B_i(x_k)=\sum_i \alpha_i(x_k)\phi_i(x_k) \quad \text{ec.(68).}$$

$$\frac{\partial E_k}{\partial B(x_k)}=\frac{1}{B(x_k)}$$

$$\frac{\partial B(x_k)}{\partial \alpha_i}=\phi_i(x_k)$$

$$\frac{\partial \alpha_i(x)}{\partial \gamma_h}=\delta_{hi}\frac{e^{\gamma_i(x)}}{\sum_j e^{\gamma_j(x)}}-\frac{e^{\gamma_i(x)}e^{\gamma_h(x)}}{(\sum_j e^{\gamma_j(x)})^2}=\alpha_i(x)(\delta_{hi}-\alpha_h(x))$$

dónde δ_{hi} vale 1 (uno) si $h=i$ y 0 (cero) si $h\neq i$.

$$\frac{\partial B(x_k)}{\partial \alpha_i}=\phi_i(x_k)$$

y para completar lo necesario para aplicar la regla de la cadena hasta las terminales de la RN se necesitan las derivadas de las funciones $\frac{\partial \phi_i(x_k)}{\partial \beta_i(x_k)}$ que dependerán de las funciones seleccionadas.

Caso kernel gaussiano

$$p_Z(Z_k)=\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}e^{-\frac{1}{2}Z^T\Sigma^{-1}Z}$$ ec.(69)

Supongamos que con un cambio de coordenadas $\boldsymbol{R_a}$, se tiene

$$p_Z(Z)=\frac{1}{(2\pi)^{1/2}\sigma}e^{-\frac{1}{2\sigma^2}(Z-\mu)^2}$$

$$\frac{\partial p_Z(Z)}{\partial \mu}=p_Z(Z)(\frac{1}{\sigma^2}(Z-\mu))$$

$$\frac{\partial p_Z(Z)}{\partial \sigma}=p(Z)(\frac{-1}{\sigma}+\frac{1}{\sigma^3}(Z-\mu)^2)$$

Caso kernel weibull

$p(x,\lambda,k)=\frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(x/\lambda)^k}$ para $x\geq 0$ y $p(x,\lambda,k)=0$ para $x<0$

16. Anexo. Aproximador lineal en espacio de caracterísctcias y regularización de Ridge.

El problema general de ajustar $y=f(x)$ en base a un conjunto de muestras, o datos: $\left[(x_i,y_i)\right]_{i=1}^N$ se puede plantear como:

$$\min_{f\in H}\left[\sum_{i=1}^{N}L(y_i,f(x_i))+\lambda J(f)\right]$$ ec.(70)

Dónde $L(.,.)$ es una función de costo a minimizar para lograr el ajuste de los datos y $J(f)$ es la función de costo que estima “la complejidad” (o falta de suavidad) de la funión f . El parámetro λ sirve para variar el peso relativo entre la regularización y el ajuste de los datos.

Entre las técnicas más usuales de aproximación de modelos, está el buscar un conjunto de funciones (generalmente llamadas características (features en inglés)) $\Phi_j(x)$ con la esperanza de poder explicar la salida como una combinación lineal de dichas características. En palabras, se trata de hacer un cambio de variables, del espacio de los datos a un espacio de características y buscar una relación lineal entre la salida y las características.

Cuando $f=\sum_{j=1}^{M}\theta_j\Phi_j(x)$ una función de regularización clásica es la de Ridge que consiste en considerar $J(f)=\sum_{j=1}^{M}\theta_j^2$. Si además, se considera $L(y_i,f(x_i))=(y_i-f(x_i))^2$, la solución a la ec.70 es simplemente la solución de mínimos cuadrados con regularización de Ridge.

Si z_i es el vector formado por las características $\Phi_j(x_i)$, el problema de la ec.70 se plantea como:

$$\min_{\theta \in R^M}\left[\sum_{i=1}^{N}(y_i - z_i^T\theta)^2 + \lambda\theta^T\theta\right] \quad \text{ec.(71)}$$

El problema 71 es convexo (por se suma de funciones de costo convexas y el dominio R^M es convexo). Por lo tanto en el óptimo se cumple que una variación $\delta\theta$ tiene que causar una variación nula en la función de costo $\delta C = 0$

$$\sum_{i=1}^{N}(y_i - z_i^T(\theta+\delta\theta))^2 + \lambda(\theta+\delta\theta)^T(\theta+\delta\theta)$$

$$C+\delta C = \sum_{i=1}^{N}(y_i - z_i^T\theta - z_i^T\delta\theta)^2 + \lambda(\theta^2 + 2\theta^T\delta\theta + \delta\theta^T\delta\theta)$$

$$C+\delta C = \sum_{i=1}^{N}((y_i - z_i^T\theta)^2 - 2(y_i - z_i^T\theta)z_i^T\delta\theta + (z_i^T\delta\theta)^2) + \lambda(\theta^2 + 2\theta^T\delta\theta + \delta\theta^T\delta\theta)$$

Restando $C = \sum_{i=1}^{N}(y_i - z_i^T\theta)^2 + \lambda\theta^T\theta$ de ambos lados y despreciando los términos de segundo orden se tiene:

$$\delta C = \sum_{i=1}^{N}(-2(y_i - z_i^T\theta)z_i^T\delta\theta) + \lambda(2\theta^T\delta\theta) = 0$$

$$\delta C = \sum_{i=1}^{N}(-2(y_i - z_i^T\theta)z_i^T\delta\theta) + \lambda(2\theta^T\delta\theta) = 0$$

usando que $z_i^T\theta = \theta^T z_i$ por se el resultado un escalar, podemos escribir:

$$\sum_{i=1}^{N}(-y_i + \theta^T z_i)z_i^T + \lambda\theta^T = 0$$

$$\sum_{i=1}^{N}(-y_i z_i^T + \theta^T z_i z_i^T) + \lambda\theta^T = 0$$

$$\sum_{i=1}^{N}(-y_i z_i^T + \theta^T z_i z_i^T) + \lambda\theta^T = 0$$

Transponiendo y despejando:

$$(\sum_{i=1}^{N} z_i z_i^T + \lambda I)\theta = \sum_{i=1}^{N} z_i y_i$$

$$\theta = (\sum_{i=1}^{N} z_i z_i^T + \lambda I)^{-1}\sum_{i=1}^{N} z_i y_i$$

16.1. Implementación de aproximadores por Kernels.

Se implementó una Clase (tipo de objeto Pascal) que da el soporte de la función de costo futuro como una Clase refinada de las ya existentes en la plataforma SimSEE para ese propósito. De esta forma, la utilización del nuevo tipo de aproximador es simple y directa en la plataforma. A la

Clase definida hay que enseñarle cómo calcular el valor de la función CF(X,k) en cualquier punto (X,k) y a calcular su gradiente en el mismo punto.

La estructura de aproximación es:

$$CF(X,k)=\sum_{j=1}^{N_K}\lambda_{kj}K(X_{kj},X,\theta_{kj}) \qquad \text{ec.(72)}$$

Donde N_K

Conclusión sobre la utilización de kernels para representar la función de Valor. Si el problema de aproximación estuviese en el cálculo de los parámetros de la estructura de aproximación, el planteo de aproximación con kernels tendría una ventaja frente a otros métodos por tener una resolución directa; pero la realidad es que no es ese nuestro problema. Nuestro problema está en tener una estructura de aproximación capaz de captar las singularidades de la función de valor en el espacio de estado, aunque ello implique tener que recurrir a métodos de calibración más complejos, o de convergencia no garantizada como es el SGD. Un ejemplo claro de lo anterior es que si utilizamos kernels polinómicos de grado 2 (como en la ref. [ ]), nuestra función de valor se aproximará por un kernel polinómico de grado 2 (la suma de de muchos kernels polinómicos de grado 2 es un kernel polinómico de grado 2); cuando la verdadera función de valor puede no adaptarse bien a esa estructura.

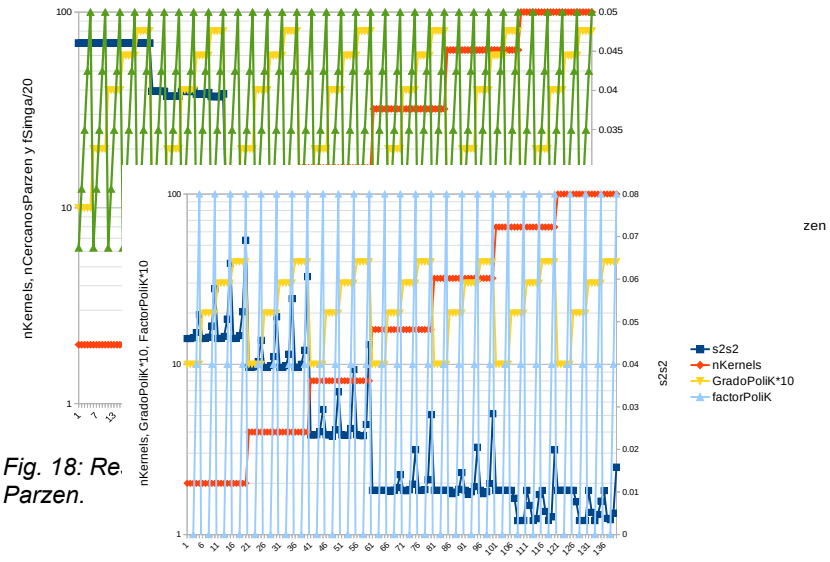
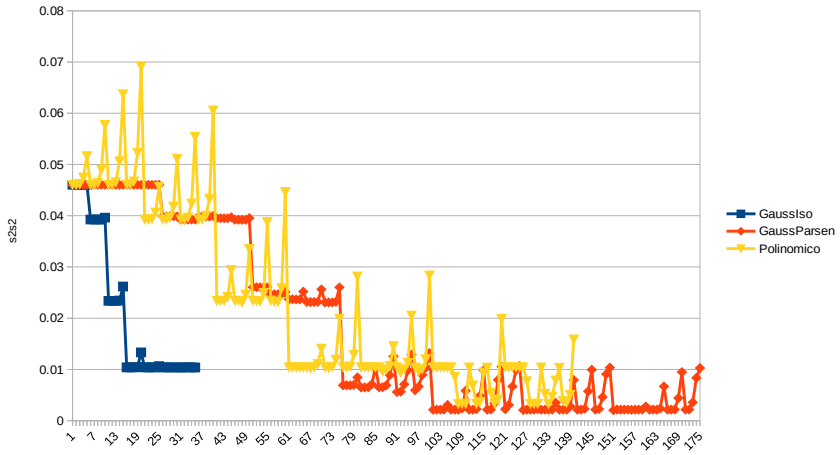


Fig. 18: Re
Parzen.

Fig. 19: Resultados con Kernels Polinómicos.


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
s2s2
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145
151
157
163
169
175
GaussIso
GaussParsen
Polinomico

17. Glosario.

En el presente documento, las palabras en mayúsculas tienen el significado que se les da en el siguiente glosario extendiendo la definición al plural. Las letras en mayúsculas entre paréntesis indican la abrebiación utilizada.

Administración del Mercado Eléctrico (ADME). Se refiere a la institución que administra el mercado de generación del Uruguay cumpliendo tanto las funciones de operador imponiendo un despacho centralizado de mínimo costo como de liquidación de las transacciones económicas del mercado.

Costo Futuro (CF). Se refiere a valor esperado del costo de la operación futura del SE a partir de un estado e instante de tiempo dados aplicando una PO dada.

Demandas. Consumidores de energía eléctrica. Cuando se habla de La Demanda del SE se está hablando de la suma de todas las Demandas del SE.

Generadores. Centrales de generación de energía eléctrica.

Política de Operación (PO). Se refiere a una función que determina un vector de control del Sistema (por ej. la potencia generada por cada generador) para cada instante de tiempo y Estado del Sistema.

Sistema de Energía (SE). Se refiere al conjunto de Generadores capaces de entregar energía eléctrica a la red, el conjunto de Demandas capaces de consumir la energía y a la red de Transmisión que los interconecta.

Sistema Dinámico (SD). Se refiere a un sistema en el que las acciones del pasado condicionan las posibilidades de acciones y/o los costos de operación presentes y futuros. Puede ser Estocástico o Determinístico.

Sistema Dinámico Estocástico (SDE). Se refiere a un SD con entradas aleatorias.

Sistema Interconectado Nacional (SIN). Se refiere al conjunto de generadores, Demandas y líneas de transmisión de un país.

Estado del Sistema. Se refiere al vector de información mínima que captura lo relevante del pasado de un SD para determinar su comportamiento futuro si se conocen las entradas futuras del SD.

Espacio de Estado. Se refiere a el espacio, en el sentido algebraico de “espacio vectorial”, en el que se define el vector que representa el Estado del Sistema.

Ecuación de Evolución de Estado. Se refiere a un modelo matemático (ecuación) que permita calcular el gradiente del vector de estado en un instante dado a partir de conocer el Estado y las entradas al sistema en ese mismo instante. En su versión de tiempo discreto, la Ecuación de Evolución de Estado permite calcular el estado al final de un Paso de Tiempo dado si se

conoce el estado al inicio del paso de tiempo y las entradas al sistema en el mismo paso de tiempo.

Función de Valor del Estado. El el valor esperado del costo futuro actualizado de la operación del sistema. Ver Costo Futuro.

Horizonte de Optimización. Es el horizonte temporal sobre el que se ejecuta el Bucle de Aprendizaje y para el cuál se va aprendiendo una Política de Operación.

Horizonte de Pronóstico. Se refiere al horizonte temporal en el cuál se dispone de pronósticos de alguno de los procesos estocásticos involucrados, por ejemplo es común disponer de buenos pronósticos de caudales afluentes a las centrales hidroeléctricas para los siguientes 10 días y de la generación eólica y solar para los siguientes siete días. También de la temperatura de los siguientes días que tiene importancia en la generación del pronóstico de Demanda. La información de pronósticos se asimila en los modelos estocásticos imponiendo sesgos y atenuadores sobre los modelos calibrados en base a la estadística de las series históricas de forma de reflejar la información adicional (al juego de correlaciones captable en las series históricas) aportada por la fuente de pronóstico. La Política de Operación Óptima puede variar por efecto del conocimiento de un pronósticos.

Horizonte de Simulación. Es el horizonte temporal, sobre el que se realiza una simulación de la operación del sistema con una Política de Operación dada. Para que sea posible la simulación, la Política de Operación debe abarcar un Horizonte de Optimización que incluya al Horizonte de Simulación. Durante el Bucle de Aprendizaje, las simulaciones se realizan con un Horizonte de Simulación igual al Horizonte de Optimización.

Paso de Tiempo. Se refiere al intervalo de tiempo o etapa en que se subdivide el Horizonte de Simulación para llevar a cabo la simulación del sistema aplicando la Ecuación de Evolución de Estado.

Transmisión. Se refiere al sistema de transporte en alta tensión que permite transportar la energía desde los Generadores a las Demandas.

18. Referencias.

[1] *SimSEE*. (2022). IIE-FING-UdeLaR. [Online]. Available: https://simsee.org
[2] R. Bellman, *Dynamic programing*. Princeton University Press, 1957.
[3] L. Weaver and N. Tao, “The Optimal Reward Baseline for Gradient-Based Reinforcement Learning,” Jan. 10, 2013, *arXiv*: arXiv:1301.2315. Accessed: Nov. 12, 2022. [Online]. Available: http://arxiv.org/abs/1301.2315
[4] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, “Variance Reduction for Reinforcement Learning in Input-Driven Environments,” Feb. 27, 2019, *arXiv*: arXiv:1807.02264. Accessed: Sep. 19, 2022. [Online]. Available: http://arxiv.org/abs/1807.02264
[5] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” 2017.

[6] A. Castellano, C. Martínez, P. Monzón, J. Andrés Bazerque, A. Ferragut, and F. Paganini, “Quadratic approximate dynamic programming for scheduling water resources: a case study,” in *2020 IEEE PES Transmission Distribution Conference and Exhibition - Latin America (T D LA)*, 2020, pp. 1–6. doi: 10.1109/TDLA47668.2020.9326171.
[7] L. Prechelt, “Early Stopping - But When?,” in *Neural Networks: Tricks of the Trade*, vol. 1524, G. B. Orr and K.-R. Müller, Eds., in Lecture Notes in Computer Science, vol. 1524. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69. doi: 10.1007/3-540-49430-8_3.
[8] N. Sato, Y. Fukuyama, T. Iizaka, and T. Matsui, “A Correntropy Based Artificial Neural Network using Early Stopping for Daily Peak Load Forecasting,” in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Chiang Mai, Thailand: IEEE, Sep. 2020, pp. 581–586. doi: 10.23919/SICE48898.2020.9240336.
[9] J. Kleijnen, A. Ridder, and R. Rubinstein, “Variance Reduction Techniques in Monte Carlo Methods,” Nov. 2010, doi: 10.2139/ssrn.1715474.
[10] W. B. Powell, *Approximate dynamic programming*. Wiley, 2011.
[11] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.
[12] F. Maciel, R. Terra, and R. Chaer, “Economic impact of considering El Niño-Southern Oscillation on the representation of streamflow in an electric system simulator: ECONOMIC IMPACT OF CONSIDERING ENSO IN AN ELECTRIC SYSTEM SIMULATOR,” *Int. J. Climatol.*, vol. 35, no. 14, pp. 4094–4102, Nov. 2015, doi: 10.1002/joc.4269.
[13] G. Flieller and R. Chaer, “Introduction of ensemble based forecasts to the electricity dispatch simulator SimSEE,” in *2020 IEEE PES Transmission & Distribution Conference and Exhibition - Latin America (T&D LA)*, Montevideo, Uruguay: IEEE, Sep. 2020, pp. 1–6. doi: 10.1109/TDLA47668.2020.9326141.
[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition. in Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.
[15] M. V. F. Pereira and L. M. V. G. Pinto, “Multi-stage stochastic optimization applied to energy planning,” *Mathematical Programming*, vol. 52, no. 1–3, pp. 359–375, May 1991, doi: 10.1007/BF01582895.
[16] S. Sethi and G. Sorger, “A theory of rolling horizon decision making,” *Ann Oper Res*, vol. 29, no. 1, pp. 387–415, Dec. 1991, doi: 10.1007/BF02283607.
[17] L. Glomb, F. Liers, and F. Rösel, “A rolling-horizon approach for multi-period optimization,” *European Journal of Operational Research*, vol. 300, no. 1, pp. 189–206, Jul. 2022, doi: 10.1016/j.ejor.2021.07.043.
[18] J. Silvente, G. M. Kopanos, E. N. Pistikopoulos, and A. Espuña, “A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids,” *Applied Energy*, vol. 155, pp. 485–501, Oct. 2015, doi: 10.1016/j.apenergy.2015.05.090.