UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE INGENIERÍA

UNIDAD DE POSGRADO E INVESTIGACIÓN

“Implementación en hardware de un módulo de predicción de generación de potencia en sistemas eólicos”

PRESENTADO POR:

ING.MEC. JORGE HUMBERTO SALAS SANCHEZ

EN OPCIÓN AL GRADO DE

MAESTRO EN INGENIERÍA

OPCIÓN ENERGÍAS RENOVABLES

MÉRIDA, YUCATÁN, MÉXICO

2 0 2 0

Autorización de impresión

“Aunque este trabajo hubiere servido para el Examen de Grado y hubiera sido aprobado por el sínodo, sólo el autor es responsable de las doctrinas emitidas en él.”

Agradecimiento

Se agradece al CONACYT por haberme seleccionado como becario y el apoyo económico que implico dicha selección.

Abstract

In this thesis work, the design of a power generation prediction module in wind systems is presented, based on neural networks and comparing its performance with the differential evolution algorithm.

The final objective of this thesis is to predict the power generated by a wind system in the short term, and to implement the algorithm in MATLAB and Texas Instrument C2000 cards.

When comparing the algorithms implemented in hardware, similar results were obtained, however, it was observed that the artificial neural network performed better in both training and evaluation. In training, the artificial neural network obtained an R correlation of 0.9733, while the differential evolution obtained 0.9640, likewise, in the evaluation, the ANN obtained an R correlation of 0.9620, and the ED one of 0.9585.

Resumen

En este trabajo de tesis se presenta el diseño de un módulo de predicción de generación de potencia en sistemas eólicos, basándose en redes neuronales y comparando su desempeño con el algoritmo de evolución diferencial.

El objetivo final de esta tesis es predecir la potencia que genera un sistema eólico a corto plazo, e implementar el algoritmo en MATLAB y en tarjetas Texas Instrument C2000.

Al comparar los algoritmos implementados en hardware se obtuvieron resultados similares, sin embargo, se observó que la red neuronal artificial obtuvo un mejor desempeño tanto en el entrenamiento como en la evaluación. En el entrenamiento, la red neuronal artificial obtuvo una correlación R de 0.9733, mientras que la evolución diferencial obtuvo 0.9640, de igual forma, en la evaluación, la RNA obtuvo una correlación R de 0.9620, y la ED una de 0.9585.

Índice

Abstract 5
Resumen 6
Capítulo 1. Introducción 11
Antecedentes 11
Planteamiento del problema 12
Objetivo general 14
Objetivos específicos 14
Capítulo 2. Marco teórico 16
Energía eólica 16
Generación de potencia a partir de energía eólica 17
Desventajas de la energía eólica 18
Ventajas de la energía eólica 21
Evaluación del potencial eólico 22
Redes neuronales artificiales 32
Componentes de una RNA 32
Modelos de una RNA 34
Algoritmos de aprendizaje 40
Pronósticos 42
Técnicas de pronósticos 42
Exactitud 43
Medidas de error agregado 43
Limitaciones de los pronósticos 45
Redes neuronales sin computadora 45
Especificaciones del hardware 46
Desempeño de las RNA sin computadora 46
Tipos de implementación de una RNA 47
Implementaciones de las RNA sin computadora 48
Procesadores digitales de señales 48
Desempeño de un DSP 50
Comparación de aplicaciones 51
Evolución diferencial 52
Mutación 53
Cruce 54

Selección 55
Estado del arte 55
Capítulo 3. Metodología 61
Implementación en hardware 67
Capítulo 4. Resultados 69
Predicción utilizando evolución diferencial 69
Predicción utilizando la red neuronal 70
Comparación de los algoritmos de predicción 71
Capítulo 5. Conclusiones 75
Recomendaciones para trabajos futuros 75
Referencias 77
Apéndice A 84
Apéndice B 86
Apéndice C 88
Código Linker 96
Código del bloque MATLAB Function para la ED 97
Código del bloque MATLAB Function para la RNA 97

Lista de figuras

Figura 1. Capacidad de energía eólica global instalada acumulada. 11
Figura 2 Estimación de la producción eléctrica mundial. 13
Figura 3 Parque eólico en Xinjiang, China. 16
Figura 4 Anemómetro de tres copas. 22
Figura 5 Anemómetro de veleta. 24
Figura 6 Diagrama de cable caliente. 25
Figura 7 Dibujo de un anemómetro laser. La luz laser es emitida (1) a través de los lentes frontales (6) del anemómetro y es retrodispersada por las moléculas de aire (7). 26
Figura 8. Anemómetro ultrasónico. 27
Figura 9 a) Anemómetro de dos rutas. b) Anemómetro de tres rutas. 28
Figura 10 Anemómetro de resonancia acústica. 29
Figura 11 Anemómetro de tubo. 30
Figura 12 Tubo de pitot estático. 31
Figura 13 Representación clásica de una red neuronal artificial. 32
Figura 14. Dependencias de una RNA. 35
Figura 15 Dos representaciones separadas de una RNA recurrente. 36
Figura 16 Categorías de hardware para redes neuronales según Heemskerk, 1995. 47

Figura 17 Ejemplo de una función costo de dos dimensiones, mostrando sus líneas de contorno y el proceso de generar $v_{i,G+1}$. 53
Figura 18 Se muestra el proceso de cruce con D=7 parámetros. 54
Figura 19 Diagrama de una tarjeta de medición de potencia (de la Cruz May 2018). 62
Figura 20 Tarjeta de medicion de potencia ya manufacturada (de la Cruz May 2018). 62
Figura 21 Historial de potencia utilizada para el entrenamiento. 64
Figura 22 Historial de potencia utilizado para la evaluación. 64
Figura 23 Estructura de la red neuronal utilizada. 65
Figura 24 Tarjeta F28379D LaunchPad de Texas Instruments. 67
Figura 25 Diagrama de flujo de la RNA implementada en hardware. 68
Figura 26 Resultado del entrenamiento de la evolución diferencial. 69
Figura 27 Resultado de la evaluación de la evolución diferencial. 69
Figura 28 Resultado del entrenamiento de la red neuronal. 71
Figura 29 Evaluación de la red neuronal. 71
Figura 30 Resultados del entrenamiento de la ED, acercamiento a un día. 72
Figura 31 Resultados del entrenamiento de la RNA, acercamiento a 1 día. 72
Figura 32 Evaluación de la ED, acercamiento a 1 día. 73
Figura 33 Evaluacion de la RNA, acercamiento a 1 día. 73
Figura 34 Se selecciona el modelo del hardware a utilizar. 88
Figura 35 Software necesario para programar el Hardware. 89
Figura 36 Selecciona la carpeta de instalación de Control Suite. 89
Figura 37 Selección de la carpeta de instalación de code composer studio. 90
Figura 38 Selección de compilador. Valor predefinido. 90
Figura 39 Resumen de la configuración hecha. 91
Figura 40 Modelo construido en simulink. 91
Figura 41 Ventana de configuración del bloque “To Workspace”. 92
Figura 42 En Data Import/Export se selecciona únicamente output en un arreglo. 92
Figura 43 Se selecciona la tarjeta apropiada y expandimos el menú de recursos del hardware. 93
Figura 44 En el menú de recursos del hardware se selecciona usar linker personalizado 93
Figura 45 Se selecciona modo externo dentro del menú de recursos. 94
Figura 46 Se selecciona el puerto de comunicaciones que utiliza la tarjeta. 94
Figura 47 Seleccionar “administrador de dispositivos” al dar click secundario en inicio. 95
Figura 48 Expandir el árbol de los puertos y encontrar el que usa la tarjeta. 95
Figura 49 Modificar el tamaño del heap_size para poder enviar todos los datos. 96

Lista de tablas

Tabla 1 Errores escalados. 43
Tabla 2 Errores porcentuales. 44
Tabla 3 Errores en tendencia para pronósticos. 44
Tabla 4 Parámetros utilizados en el modelo de la red neuronal. 65
Tabla 5 Parámetros utilizados en el algoritmo de evolución diferencial. 66
Tabla 6 Pesos obtenidos con evolucion diferencial 70
Tabla 7 Errores estadísticos de ambos algoritmos en el entrenamiento. 74
Tabla 8 Errores estadísticos de ambos algoritmos en la evaluación. 74

Capítulo 1. Introducción

Antecedentes

El aprovechamiento del potencial energético del viento ha evolucionado a través de la historia de la humanidad por al menos 3000 años (Sathyajith 2006). Durante las últimas décadas ha experimentado una tasa de crecimiento sin precedentes debido a la transición hacia el desarrollo sustentable, provocado por la crisis del petróleo de 1973 y 1979, el descubrimiento del agujero de la capa de ozono causado por los clorofluorocarbonos (Molina y Rowland 1974) y el calentamiento global provocado por los gases de efecto invernadero (Broecker 1975). Esta tendencia ha sido impulsada por el desarrollo de una conciencia ecológica y las acciones internacionales como el Protocolo de Kyoto en 1997 y 2005. Entre 2010 y 2015 la capacidad de viento instalada acumulada global ha crecido un 17% anual en promedio, como se ve en la Figura 1 (GWEC 2015).

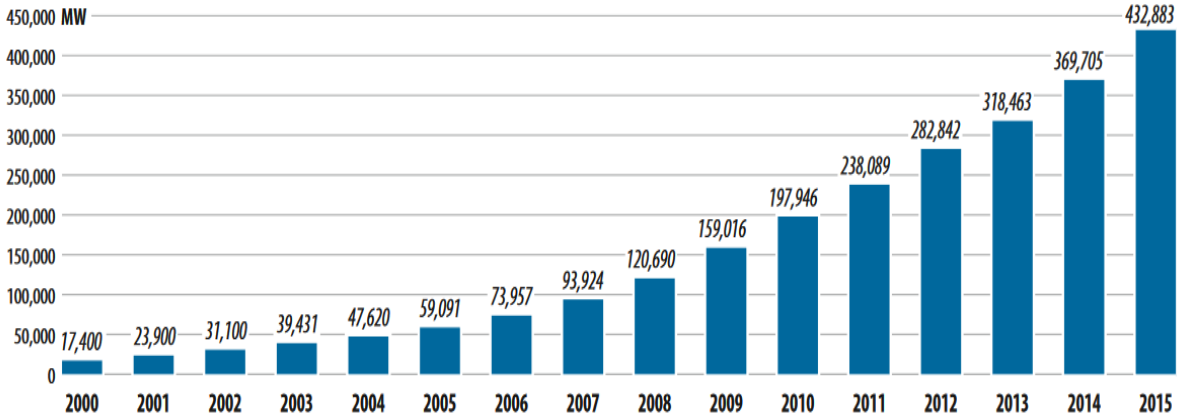

Figura 1. Capacidad de energía eólica global instalada acumulada.

En México se ha impulsado fuertemente al sector industrial y al energético para que usen energías renovables. Entre 2005 y 2015, la energía eólica es la tecnología que más ha crecido en capacidad instalada en México: 105%. Actualmente, México está en el lugar número 18 como generador de energía eólica a nivel mundial, y en el segundo puesto en América Latina. El país cuenta con una vocación eólica natural gracias a que, existen zonas con excelentes corrientes de viento, como “el corredor del viento”, el cual ha alcanzado una capacidad instalada mayor a los 2GW (Ávila-Calero 2017). La energía eólica contribuye con cerca del 6% de la energía que requiere el país. Actualmente en México se tienen 4,000 MW de capacidad instalada generados a través de las más de 2,000 turbinas que operan en los

diferentes parques Eólicos. El Estado de Oaxaca es el que más energía Eólica genera al contar con 1,500 turbinas en operación, por todo esto, se le considera a México como una de las áreas más prometedoras para el desarrollo de la energía eólica (Soler-Bietz 2009).

Para mejorar la confiabilidad de los sistemas eólicos, se ha realizado una intensa investigación sobre la predicción del viento, basada en análisis numéricos, aproximaciones estadísticas, métodos de inteligencia artificial y métodos híbridos (Wu y Ilong 2007). Los análisis numéricos dependen del terreno y las curvas de potencia de los generadores para construir un modelo matemático complejo, estos enfoques no son adecuados para aplicaciones de corto plazo y dependen de condiciones atmosféricas estables. Las aproximaciones estadísticas tienen la ventaja de ser de bajo costo ya que solo necesitan datos históricos, por otra parte, la precisión de la predicción se reduce para largos horizontes de tiempo. Los métodos de inteligencia artificial son más apropiados para predicciones a corto plazo, estos métodos se basan en series de tiempo para construir un modelo que aproxime una relación entre los datos de entrada y los de salida. (Alanis 2009).

Planteamiento del problema

La energía eólica es vista como una de las formas más efectivas para mitigar el cambio climático (Soler-Bietz 2009). Su capacidad global ha crecido 12% en 2016, con 55,000 MW de nueva capacidad instalada mundialmente. Sin tomar en cuenta la hidroelectricidad, la energía eólica es la energía renovable líder aportando el 3.7% de la producción de electricidad mundial, seguido por la bioelectricidad con 2%, esto se observa en la Figura 2 (REN21 2017).

Para hacer un uso eficiente y correcto de la energía eólica es necesario saber sus características, persistencia, disponibilidad, variación y la predicción de la velocidad (Kalogirou 1999). La predicción de la velocidad del viento es necesaria para el inventario de energía eólica que es de gran importancia para la administración del recurso eólico (Bechrakis y Sparis 2004). Además, predecirla permite la programación de conexión o desconexión de la turbina, planear el mantenimiento y administrar las cargas, lo que resulta en una mejorar de la eficiencia del sistema (Androutsos et al. 1994). Para la industria es necesaria la predicción de los parámetros meteorológicos como la velocidad del viento y la radiación solar para

propósitos de diseño, análisis de desempeño y estimaciones de costos. Predecir la velocidad del viento o la potencia generada no es una tarea fácil ya que tiene una naturaleza estocástica con una alta tasa de cambio. Las series de tiempo de velocidades de viento presentan un comportamiento aleatorio, por lo que, para tener un buen sistema de predicción es necesario garantizar la fiabilidad del sistema (Senjy et al. 2016).

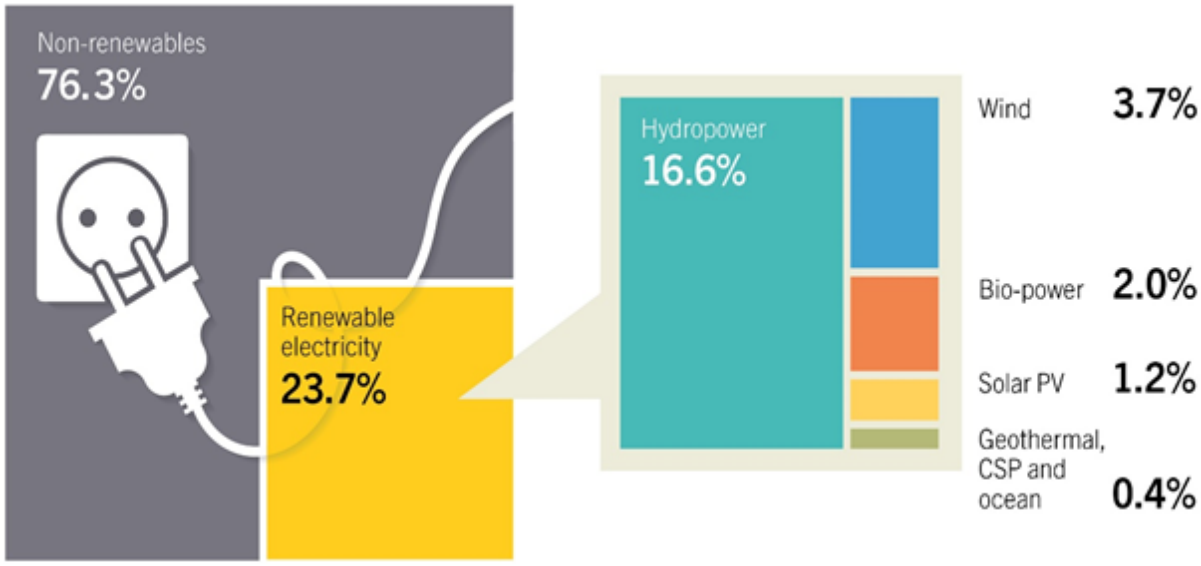

Figura 2 Estimación de la producción eléctrica mundial.

Se han implementado predicciones de corto plazo de energía eólica con redes neuronales de alto orden y han superado a otros métodos debido a su rápido algoritmo de aprendizaje y su versatilidad para variar el horizonte de predicción (Kariniotakis et al. 1996).

Las redes neuronales implementadas en hardware son importantes para la industria ya que ofrecen bajo consumo energético y son de un tamaño más pequeño que las implementadas en software de computadora, además, pueden ser embebidas en una amplia variedad de sistemas. Existen bibliotecas de software para modelos tradicionales de redes neuronales artificiales (MATLAB). Sin embargo, realizar algoritmos fuera de estas bibliotecas permite personalización y optimización del código. Otra ventaja de las redes neuronales basadas en hardware es que pueden aprovechar las operaciones en paralelo de las neuronas y conseguir altas velocidades en el procesamiento de información (Kariniotakis et al. 1996).

Objetivo general

Implementar en hardware un método de inteligencia computacional que prediga el comportamiento del recurso eólico.

Objetivos específicos

1. Implementar la red neuronal en hardware.
2. Evaluar el desempeño de la inteligencia computacional en hardware.
3. Evaluar el desempeño de otro algoritmo de predicción en hardware y compararlos.

Al lograr los objetivos se conocerá cual es el mejor método de predicción de generación de potencia basándose en los errores RMSE, MSE y la correlación R, con ello, se podrán programar mantenimientos cuando se espere una baja generación de potencia y cuando se espere una alta generación se tendrán listos los aerogeneradores. Además, si se cuenta con un sistema de almacenamiento de energía, con ayuda de las predicciones se podría tener una mejor administración de está.

Este trabajo se organiza de la siguiente manera, en la introducción se ha presentado una visión general, la cual está dividida en antecedentes, planteamiento del problema y objetivos. Posteriormente, en el capítulo 2, se presenta el marco teórico con los conceptos fundamentales utilizados en el trabajo, a partir de una amplia revisión de la literatura. Este capítulo contiene la teoría y conceptos del trabajo, así como las técnicas a utilizar para alcanzar los objetivos además del estado del arte en los métodos de solución del problema, se tratan temas como la energía eólica, redes neuronales artificiales, técnicas de pronósticos, redes neuronales sin computadora, procesadores digitales de señales y algoritmos de evolución diferencial. En el capítulo 3 se aborda la metodología, donde se representa como se realizó el trabajo de investigación, para lo cual se describen los procedimientos empleados, métodos utilizados y experimentos realizados. A grandes rasgos la metodología consiste en implementar los algoritmos de entrenamiento en MATLAB, posteriormente se implementan los algoritmos ya entrenados en el hardware para obtener predicciones empleando datos no utilizados en el entrenamiento y se comparan los resultados de cada algoritmo. El

capítulo 4 revisa los resultados obtenidos de la investigación al seguir los procedimientos del capítulo anterior. Y, por último, en el capítulo 5 se presentan las conclusiones donde se interpreta la información obtenida y se realizan recomendaciones para trabajos futuros.

Capítulo 2. Marco teórico

Energía eólica

La energía eólica es energía que se obtiene del viento o, dicho de otro modo, es el aprovechamiento de la energía cinética de las masas de aire que puede convertirse en energía mecánica y a partir de ella en electricidad u otras formas útiles de energía para las actividades humanas. La energía eólica es un recurso abundante, renovable y limpio que ayuda a disminuir las emisiones de gases de efecto invernadero al reemplazar fuentes de energía a base de combustibles fósiles. El impacto ambiental de este tipo de energía es, generalmente, menos problemático que el de otras fuentes de energía.

Figura 3 Parque eólico en Xinjiang, China.

En la actualidad, la energía eólica se utiliza principalmente para producir electricidad, lo que se consigue mediante aerogeneradores conectados a las grandes redes de distribución de energía eléctrica. Los parques eólicos construidos en tierra, como se muestran en la Figura 3, suponen una fuente de energía cada vez más barata y competitiva, e incluso más barata en muchas regiones que otras fuentes de energía convencionales (Walwyn y Brent 2015). Las granjas eólicas tienen un impacto visual y típicamente necesitan más superficie que otras fuentes

de energía (Nathan et al. 2015), además necesitan ser construidas en áreas rurales o naturales, ya que se suele tener mejores velocidades de viento en estas áreas, lo que puede ocasionar la industrialización del campo (Szarka 2007) y la perdida de hábitat. Pequeñas granjas eólicas pueden proporcionar electricidad a regiones aisladas que no tienen acceso a la red eléctrica mediante instalaciones eólicas de tamaño reducido. Las compañías eléctricas distribuidoras adquieren cada vez en mayor medida el excedente de electricidad producido por pequeñas instalaciones eólicas domésticas (Gipe 1993).

La energía del viento es bastante estable y predecible a escala anual, aunque presenta variaciones significativas a escalas de tiempo menores. Al incrementarse la proporción de energía eólica producida en una determinada región o país, se hace imprescindible establecer una serie de mejoras en la red eléctrica local (Hannele et al. 2016). Diversas técnicas de control energético, como una mayor capacidad de almacenamiento de energía, una distribución geográfica amplia de los aerogeneradores, la disponibilidad de fuentes de energía de respaldo, la posibilidad de exportar o importar energía a regiones vecinas o la reducción de la demanda cuando la producción eólica es menor, pueden ayudar a mitigar en gran medida estos problemas (Armaroli y Balzani 2011). Además, son de extrema importancia las previsiones de producción eólica que permiten a los gestores de la red eléctrica estar preparados y anticiparse frente a las previsibles variaciones en la producción eólica que puedan tener lugar a corto plazo (Platt et al. 2012; Huang et al. 2014).

Generación de potencia a partir de energía eólica

La energía del viento está relacionada con el movimiento de las masas de aire que se desplazan desde zonas de alta presión atmosférica hacia zonas adyacentes de menor presión, con velocidades proporcionales al gradiente de presión.

Los vientos se generan a causa del calentamiento no uniforme de la superficie terrestre debido a la radiación solar; entre el 1 y el 2 % de la energía proveniente del Sol se convierte en viento (Lorenz, 1967). Durante el día, los continentes transfieren una mayor cantidad de energía solar al aire que las masas de agua, haciendo que el aire se caliente y se expanda, por lo que se vuelve menos denso y

se eleva. El aire más frío y pesado que proviene de los mares, océanos y grandes lagos se pone en movimiento para ocupar el lugar dejado por el aire caliente.

Para poder aprovechar la energía eólica es importante conocer las variaciones diurnas, nocturnas y estacionales de los vientos, la variación de la velocidad del viento con la altura sobre el suelo, la velocidad de las ráfagas en breves periodos de tiempo, y los valores máximos ocurridos en series históricas de datos con una duración mínima de 20 años. Para poder utilizar la energía del viento, es necesario que este alcance una velocidad mínima que depende del aerogenerador que se vaya a utilizar pero que suele empezar entre los 3 m/s (10 km/h) y los 4 m/s (14,4 km/h), velocidad llamada “cut-in speed”, y que no supere los 25 m/s (90 km/h), velocidad llamada “cut-out speed”.

La energía del viento se aprovecha mediante el uso de máquinas eólicas o aeromotores capaces de transformar la energía eólica en energía mecánica de rotación utilizable, ya sea para accionar directamente máquinas o para la producción de energía eléctrica. Este último caso es el más utilizado en la actualidad, el sistema de conversión consta de un generador eléctrico con sistemas de control y de conexión a red, en conjunto es conocido como aerogenerador. En los aerogeneradores la energía eólica mueve una hélice y mediante un sistema mecánico se hace girar el rotor de un generador, normalmente un alternador, que produce energía eléctrica. Para que su instalación resulte rentable, suelen agruparse en concentraciones denominadas parques eólicos.

Desventajas de la energía eólica

Aspectos técnicos

Como se mencionó previamente, la variabilidad natural y la predictibilidad del viento es la principal desventaja para que la energía eólica pueda ser usada como única fuente de energía eléctrica, para lograrlo es necesario almacenar la energía que se produce cuando hay viento para poder luego utilizarla cuando no lo hay. Pero hasta el momento no existen sistemas lo suficientemente grandes como para almacenar cantidades considerables de energía de forma eficiente. Por lo tanto, para hacer frente a los valles en la curva de producción de energía eólica y evitar apagones generalizados, es indispensable un respaldo de las energías convencionales como centrales termoeléctricas de carbón, gas natural, petróleo o ciclo combinado o centrales hidroeléctricas reversibles, por ejemplo. Esto supone un inconveniente,

puesto que cuando respaldan a la eólica, las centrales de carbón no pueden funcionar a su rendimiento óptimo. Tienen que quedarse muy por debajo de su óptimo para poder subir sustancialmente su producción en el momento en que amaine el viento. Es por lo que, cuando funcionan en este modo, las centrales térmicas consumen más combustible por kWh producido (Belderbos y Delarue 2015). También, al aumentar y disminuir su producción cada vez que cambia la velocidad del viento se produce un desgaste mayor de la maquinaría (Lefton y Besuner 2006). Además, la variabilidad en la producción de energía eólica tiene otras importantes consecuencias:

- Para distribuir la electricidad producida por cada parque eólico (que suelen estar situados además en parajes naturales apartados) es necesario construir líneas de alta tensión que sean capaces de conducir el máximo de electricidad que sea capaz de producir la instalación.
- Técnicamente, uno de los mayores inconvenientes de los aerogeneradores es el llamado hueco de tensión. Ante uno de estos fenómenos, las protecciones de los aerogeneradores con motores de jaula de ardilla provocan la desconexión de la red para evitar ser dañados. Este problema se soluciona utilizando motores síncronos, aunque es bastante más fácil asegurarse de que la red a la que se va a conectar sea fuerte y estable.
- Además de la evidente necesidad de una velocidad mínima en el viento para poder mover las aspas, existe también una limitación superior: una máquina puede estar generando al máximo de su potencia, pero si la velocidad del viento sobrepasa las especificaciones del aerogenerador, es obligatorio desconectarlo de la red o cambiar la inclinación de las aspas para que dejen de girar, puesto que su estructura puede resultar dañada por los esfuerzos que aparecen en el eje. La consecuencia inmediata es un descenso evidente de la producción eléctrica, a pesar de haber viento en abundancia, y supone otro factor más de incertidumbre a la hora de contar con esta energía en la red eléctrica de consumo.

Aunque estos problemas parecen únicos a la energía eólica, son comunes a todas las energías de origen natural:

- Un panel solar solo producirá energía mientras haya suficiente luz solar.

- Una central hidroeléctrica solo podrá producir mientras las condiciones hídricas y las precipitaciones permitan la liberación de agua.

Una de las formas de mitigar la falta de control sobre los recursos renovables (viento, radiación solar), son los llamados sistemas híbridos, donde se combinan fuentes de energía junto con almacenamiento. Hay una tendencia a la creación de centrales renovables en las que participan generadores eólicos, solares y almacenamiento por baterías. En países como Australia o Estados Unidos se está regulando su uso e incluso se definen tarifas específicas para la inyección de energía desde estas centrales que comienzan a competir de igual a igual con las centrales basadas en combustibles fósiles, dado que comienzan a tener una previsión de generación a un día o más.

Aspectos medioambientales

- Generalmente, aunque no siempre, la generación eólica se combina con centrales térmicas, lo que lleva a que algunas personas consideren que realmente no se ahorran demasiadas emisiones de dióxido de carbono. No obstante, hay que tener en cuenta que ningún tipo de energía renovable permite, al menos por sí sola, cubrir toda la demanda y producción de electricidad, pero sin embargo su aportación a la red eléctrica es netamente positiva desde el punto de vista del ahorro de emisiones.
- Existen parques eólicos en México en espacios protegidos como el parque Dzilam Bravo Eólica del Golfo 1, el cual se encuentra dentro de las RTP (Regiones terrestres prioritarias) (Arriaga 2000), lo que supone un impacto natural, si bien reducido, debido a la actividad humana.
- Al comienzo de su instalación, los lugares seleccionados para ello coincidieron con las rutas de las aves migratorias o con las zonas donde las aves aprovechan vientos de ladera, lo que hace que los aerogeneradores entren en conflicto con aves y murciélagos. Afortunadamente los niveles de mortandad son muy bajos en comparación con otras causas como por ejemplo los atropellos (Calvert et al. 2013). Actualmente los estudios de impacto ambiental necesarios para el reconocimiento del plan del parque eólico tienen en consideración la situación ornitológica de la zona. Además, los aerogeneradores actuales son de baja velocidad de rotación, lo que debería reducir el problema de choque con las aves.

- El impacto paisajístico es una nota importante debido a la disposición de los elementos horizontales que lo componen y la aparición de un elemento vertical como es el aerogenerador. Además, al girar se produce una sombra intermitente. Esto, unido al ruido generado por las palas de la turbina, puede llevar a la gente hasta un alto nivel de estrés, con efectos de consideración para la salud. No obstante, la mejora del diseño de los aerogeneradores ha permitido ir reduciendo progresivamente el ruido que producen (Pedersen 2004).
- La apertura de parques eólicos y la presencia de operarios en ellos hace que la presencia humana sea constante en lugares hasta entonces poco transitados, lo que afecta también a la fauna.

Ventajas de la energía eólica

- Es un tipo de energía renovable ya que tiene su origen en procesos atmosféricos debidos a la energía que llega a la Tierra procedente del Sol.
- Es una energía limpia al no requerir una combustión, por lo que no produce emisiones atmosféricas ni residuos contaminantes, evitando así un incremento del efecto invernadero y el cambio climático.
- Puede instalarse en espacios no aptos para otros fines, por ejemplo, en zonas desérticas, próximas a la costa, en laderas áridas o muy empinadas para ser cultivables.
- Puede convivir con otros usos del suelo, por ejemplo, prados para uso ganadero o cultivos bajos como trigo, maíz, patatas, remolacha, etc.
- Crea un elevado número de puestos de trabajo en las plantas de ensamblaje y las zonas de instalación.
- Su instalación es rápida, entre 4 y 9 meses.
- Su inclusión en una red eléctrica permite, cuando las condiciones del viento son adecuadas, ahorrar combustible en las centrales térmicas y/o agua en los embalses de las centrales hidroeléctricas.
- Su utilización combinada con otros tipos de energía, habitualmente la energía solar fotovoltaica, permite la autoalimentación de viviendas, y terminando así con la necesidad de conectarse a redes de suministro.
- La dispersión geográfica de los parques eólicos, en países como España, permite compensar la baja producción de unos parques eólicos por falta de

viento con la alta producción en otras zonas. De esta forma se estabiliza la forma de onda producida en la generación eléctrica, solventando los problemas que presentaban los aerogeneradores como productores de energía en sus inicios.

- Es posible construir parques eólicos en el mar, donde el viento es más fuerte, más constante y el impacto social es menor, aunque aumentan los costos de instalación y mantenimiento.

Evaluación del potencial eólico

Para medir la velocidad del viento se utilizan unos dispositivos llamados anemómetros; existen muy diversos tipos de anemómetros y se clasifican en de velocidad y de presión.

Anemómetros de velocidad

Figura 4 Anemómetro de tres copas.

El anemómetro de copas es el más simple y fue inventado en 1845. Consiste en copas hemisféricas montadas en brazos horizontales, las cuales son montadas en un eje vertical. La Figura 4 muestra un anemómetro de copas. El flujo de aire que pasa por las copas en cualquier dirección horizontal hace girar al eje a una tasa proporcional a la velocidad del viento. Por lo tanto, al contar las vueltas del eje en

un intervalo de tiempo produce un valor proporcional a la velocidad promedio del viento. También es llamado anemómetro rotacional.

En un anemómetro con cuatro copas, al estar las copas en un arreglo simétrico, es fácil ver que el viento siempre tiene presente el hueco de una copa y la parte sólida de otra. Ya que un hemisferio hueco tiene un coeficiente de arrastre de .38 en el lado esférico y 1.42 en el lado hueco (Hoerner 1965), se genera más fuerza en el lado hueco y al existir esta fuerza asimétrica se genera esfuerzo de torsión en el eje del anemómetro, causando que gire.

Teóricamente, la velocidad de rotación del anemómetro debería ser proporcional a la velocidad del viento, porque la fuerza producida sobre un objeto es proporcional a la velocidad del fluido que fluye a su alrededor, pero otros factores influyen en la velocidad de rotación, como la turbulencia producida por el dispositivo, y la fricción del punto de montaje. Cuando el Dr. John Thomas Romney Robinson diseñó por primera vez su anemómetro, afirmó que las copas se movían un tercio de la velocidad del viento, sin verse afectadas por el tamaño de la copa o la longitud del brazo. Al parecer, esto fue confirmado por algunos primeros experimentos independientes, pero fue incorrecto. En cambio, la relación entre la velocidad del viento y la de las copas, el factor anemómetro, depende de las dimensiones de las copas y los brazos, y puede tener un valor entre dos y un poco más de tres. Todos los experimentos anteriores con un anemómetro tuvieron que repetirse después de descubrir el error.

El anemómetro de tres copas desarrollado por el canadiense John Patterson en 1926 y las posteriores mejoras de la copa por Brevoort & Joiner de los Estados Unidos en 1935 llevaron a un diseño de rueda dentada con una respuesta casi lineal y tuvo un error de menos del 3% hasta 60 mph (97 km / h). Patterson descubrió que cada copa producía un par máximo cuando estaba a 45 ° del flujo del viento. El anemómetro de tres copas también tuvo un par más constante y respondió más rápidamente a las ráfagas que el anemómetro de cuatro copas.

El anemómetro de tres copas fue modificado por el Dr. Derek Weston de Australia en 1991 para medir la dirección y la velocidad del viento. Weston agregó una etiqueta a una copa, lo que hace que la velocidad de la rueda aumente y disminuya a medida que la etiqueta se mueve alternativamente con y contra el viento. La

dirección del viento se calcula a partir de estos cambios cíclicos en la velocidad de la rueda dentada, mientras que la velocidad del viento se determina a partir de la velocidad media de la rueda dentada. Los anemómetros de tres copas se utilizan actualmente como el estándar de la industria para los estudios y prácticas de evaluación de recursos eólicos.

Los anemómetros de veleta se pueden describir como un molino de viento o un anemómetro de hélice. A diferencia del anemómetro Robinson, cuyo eje de rotación es vertical, el anemómetro de paleta debe tener su eje paralelo a la dirección del viento y, por lo tanto, horizontal. Además, dado que el viento varía en dirección y el eje tiene que seguir sus cambios, se debe emplear una veleta o algún otro dispositivo para cumplir el mismo propósito. Este anemómetro se observa en la Figura 5.

Figura 5 Anemómetro de veleta.

Por lo tanto, un anemómetro de paleta combina una hélice y una veleta en el mismo eje para obtener mediciones exactas y precisas de la velocidad y dirección del viento con el mismo instrumento. La velocidad de la hélice se mide mediante un contador de revoluciones y se convierte en una velocidad del viento mediante un circuito electrónico. Por lo tanto, el caudal volumétrico puede calcularse si se conoce el área de la sección transversal.

Los Anemómetros de cable caliente se muestran en la Figura 6 y usan un alambre fino (del orden de varios micrómetros) calentado eléctricamente a cierta temperatura por encima del ambiente. El aire que pasa por el cable enfría el cable. Como la resistencia eléctrica de la mayoría de los metales depende de la temperatura del metal (el tungsteno es una opción popular para los cables calientes), se puede obtener una relación entre la temperatura del cable y la velocidad del flujo.

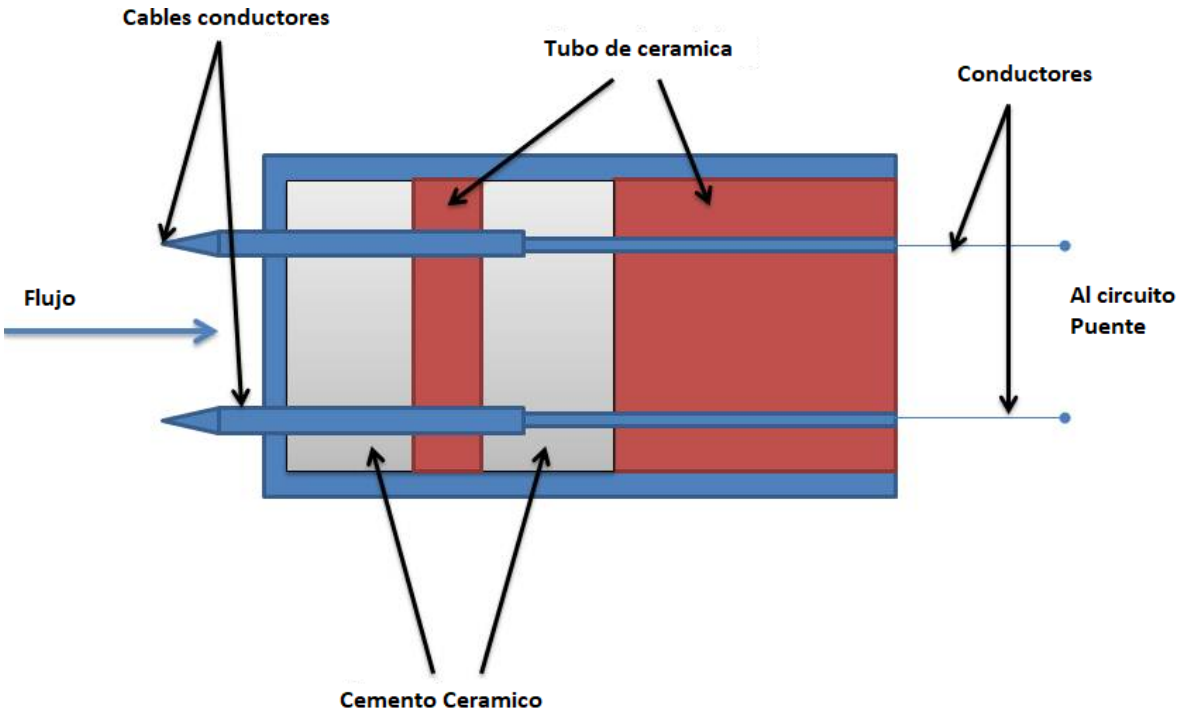

Figura 6 Diagrama de cable caliente.

Los anemómetros de alambre caliente, aunque extremadamente delicados, tienen una respuesta de frecuencia extremadamente alta y una resolución espacial fina en comparación con otros métodos de medición, y como tal se emplean casi universalmente para el estudio detallado de flujos turbulentos, o cualquier flujo en el que las fluctuaciones de velocidad rápida sean de interés.

Los anemómetros Doppler utilizan un haz de luz de un láser que se divide en dos haces, como se describe en la Figura 7, un haz es propagado fuera del anemómetro. Las partículas que fluye junto con las moléculas de aire cerca de donde sale el haz, reflejan o dispersan hacia un detector la luz que regresa, donde se mide en relación con el haz original. Cuando las partículas están en gran movimiento, producen un efecto Doppler, el cual, es usado para calcular la

velocidad de las partículas y, por lo tanto, el aire alrededor del anemómetro (Iten 1976).

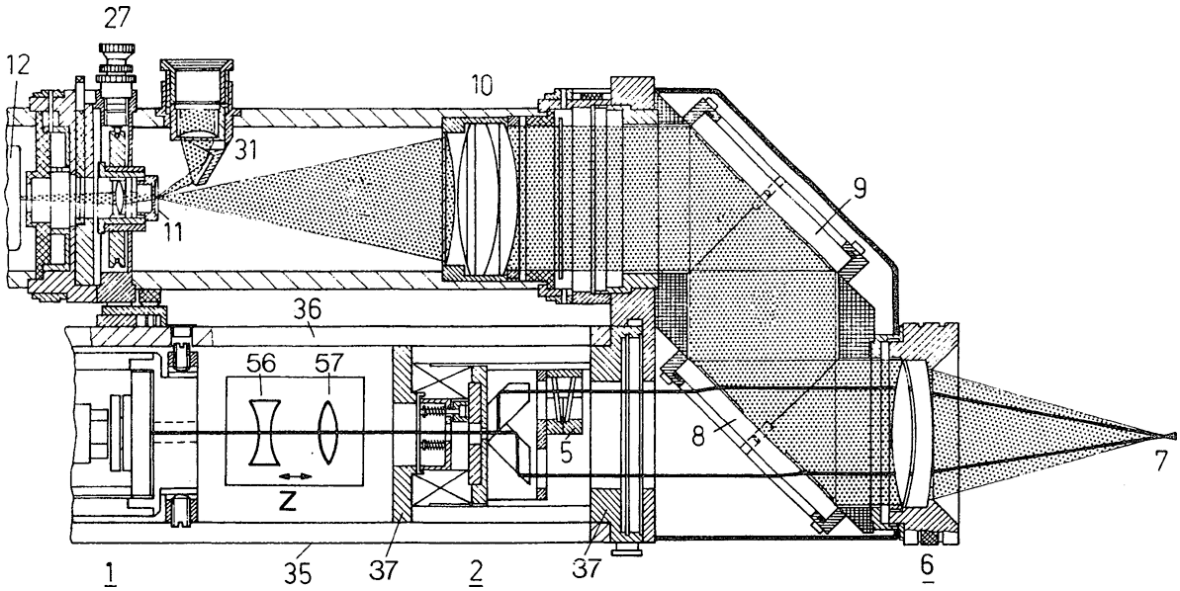

Figura 7 Dibujo de un anemómetro laser. La luz laser es emitida (1) a través de los lentes frontales (6) del anemómetro y es retrodispersada por las moléculas de aire (7).

Los anemómetros ultrasónicos, mostrados en la Figura 8, fueron desarrollados por primera vez en la década de 1950 y utilizan ondas de sonido ultrasónicas para medir la velocidad del viento en función del tiempo de vuelo de los pulsos sónicos entre pares de transductores. Las mediciones de pares de transductores se pueden combinar para obtener una medición de la velocidad en flujo de 1, 2 o 3 dimensiones. La resolución espacial viene dada por la longitud del camino entre los transductores, que suele ser de 10 a 20 cm. Los anemómetros ultrasónicos pueden tomar mediciones con una resolución temporal muy fina, 20 Hz o mejor, lo que los hace muy adecuados para mediciones de turbulencia. La falta de piezas móviles las hace apropiadas para su uso a largo plazo en estaciones meteorológicas automáticas al aire libre y en boyas meteorológicas donde la precisión y la fiabilidad de los anemómetros tradicionales de copas se ven afectadas negativamente por el aire salado o el polvo. Su principal desventaja es la distorsión del flujo de aire por la estructura que soporta los transductores, lo que requiere una corrección basada en las mediciones del túnel de viento para minimizar el efecto. Otra desventaja es la menor precisión debido a la precipitación, donde las gotas de lluvia pueden variar la velocidad del sonido.

Figura 8. Anemómetro ultrasónico.

Dado que la velocidad del sonido varía con la temperatura y es prácticamente estable con el cambio de presión, los anemómetros ultrasónicos también se utilizan como termómetros.

Los anemómetros sónicos bidimensionales (velocidad y dirección del viento) se utilizan en aplicaciones como estaciones meteorológicas, navegación de barcos, aviación, boyas meteorológicas y turbinas eólicas. El monitoreo de las turbinas eólicas generalmente requiere una frecuencia de actualización de las mediciones de la velocidad del viento de 3 Hz (Giebhardt R., Jochen T. *2010*), que se logra fácilmente mediante anemómetros sónicos.

Los sensores de viento bidimensionales son de dos tipos:

- **Dos rutas**: estos sensores tienen cuatro brazos. La desventaja de este tipo de sensor es que cuando el viento viene en la dirección de una ruta de ultrasonido, los brazos perturban el flujo de aire, reduciendo la precisión de la medición resultante. Este anemómetro se puede observar en la Figura 9 a).

- **Tres rutas**: estos sensores tienen tres brazos, como se ve en la Figura 9 b). Proporcionan una redundancia de ruta de la medición que mejora la precisión del sensor y reduce la turbulencia aerodinámica.

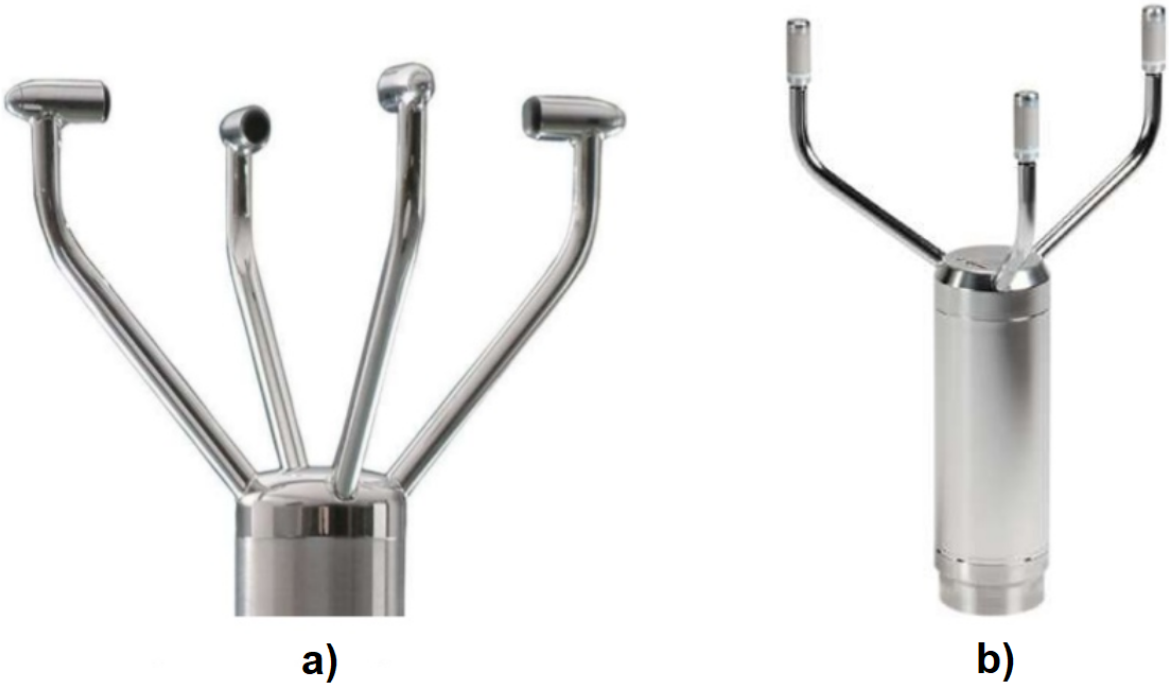

Figura 9 a) Anemómetro de dos rutas. b) Anemómetro de tres rutas.

Los anemómetros de resonancia acústica son una variante más reciente del anemómetro ultrasónico. La tecnología fue inventada por Savvas Kapartis y patentada en 1999. Mientras que los anemómetros sónicos convencionales dependen de la medición del tiempo de vuelo, los sensores de resonancia acústica utilizan ondas acústicas resonantes dentro de una pequeña cavidad.

Integrado en la cavidad hay una serie de transductores ultrasónicos, que se utilizan para crear patrones de ondas estacionarias separados a frecuencias ultrasónicas. A medida que el viento pasa a través de la cavidad, se produce un cambio en la propiedad de la onda (cambio de fase). Al medir la cantidad de desplazamiento de fase en las señales recibidas por cada transductor, y luego al procesar matemáticamente los datos, el sensor puede proporcionar una medición horizontal precisa de la velocidad y dirección del viento.

La tecnología de resonancia acústica permite la medición dentro de una pequeña cavidad, por lo tanto, los sensores tienden a ser típicamente de menor tamaño que otros sensores ultrasónicos. El pequeño tamaño de los anemómetros de resonancia

acústica los hace físicamente fuertes y fáciles de calentar y, por lo tanto, resistentes al hielo. Esta combinación de características significa que alcanzan altos niveles de disponibilidad de datos y son muy adecuadas para el control de turbinas eólicas y para otros usos que requieren sensores pequeños y robustos, como la meteorología en campo. Un problema con este tipo de sensor es la precisión de la medición en comparación con un sensor mecánico calibrado. Para muchos usos finales, esta debilidad se compensa con la longevidad del sensor y el hecho de que no requiere recalibrar una vez instalado. Este anemómetro se muestra en la Figura 10.

Figura 10 Anemómetro de resonancia acústica.

Anemómetros de presión

El anemómetro de James Lind de 1775 consistía en un tubo de vidrio en U que contenía un manómetro líquido (válvula de presión), con un extremo doblado en dirección horizontal para enfrentar el viento y el otro extremo vertical permanece paralelo al flujo del viento. Aunque el Lind no fue el primero, fue el anemómetro más práctico y mejor conocido de este tipo. Si el viento sopla en la boca de un tubo, causa un aumento de presión en un lado del manómetro. El viento sobre el extremo abierto de un tubo vertical causa pocos cambios en la presión en el otro lado del manómetro. La diferencia de elevación resultante en las dos patas del tubo en U es una indicación de la velocidad del viento. Sin embargo, una medición precisa requiere que la velocidad del viento esté directamente en el extremo abierto del tubo; Las pequeñas desviaciones de la dirección del viento provocan grandes variaciones en la lectura.

El exitoso anemómetro de tubo de presión de metal de William Henry Dines en 1892, se muestra en la Figura 11 y utilizó la misma diferencia de presión entre la boca abierta de un tubo recto frente al viento y un anillo de pequeños agujeros en un tubo vertical que está cerrado en el extremo superior. Ambos están montados a la misma altura. Las diferencias de presión de las que depende la acción son muy pequeñas, y se requieren medios especiales para registrarlas. El registrador consiste en un flotador en una cámara sellada parcialmente llena de agua. La tubería del tubo recto se conecta a la parte superior de la cámara sellada y la tubería de los tubos pequeños se dirige hacia el fondo dentro del flotador. Como la diferencia de presión determina la posición vertical del flotador, esta es una medida de la velocidad del viento.

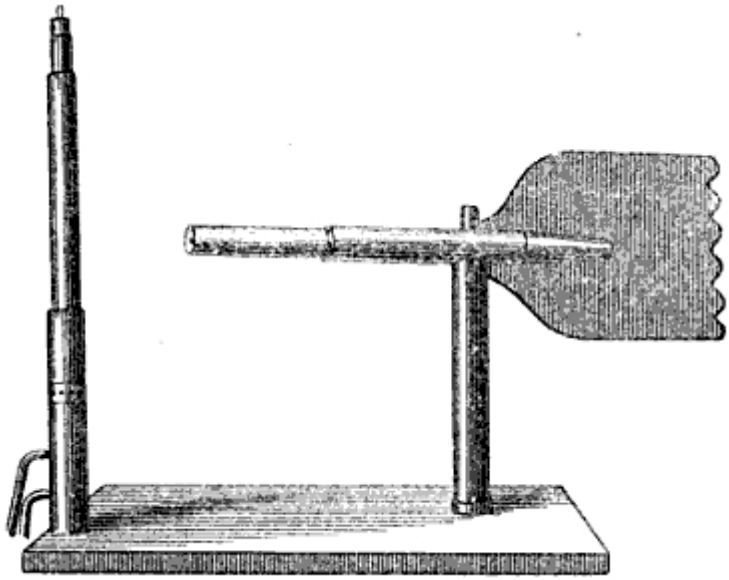

Figura 11 Anemómetro de tubo.

La gran ventaja del anemómetro de tubo radica en el hecho de que la parte expuesta se puede montar en un poste alto y no requiere lubricación ni atención durante años; y la parte de registro se puede colocar en cualquier posición conveniente. Se requieren dos tubos de conexión. A primera vista, puede parecer que una conexión serviría, pero las diferencias de presión de las que dependen estos instrumentos son tan pequeñas que debe considerarse la presión del aire en la habitación donde se coloca la parte de grabación.

Si bien el anemómetro Dines tuvo un error de solo 1% a 16 km / h, no respondió muy bien a los vientos bajos debido a la pobre respuesta de la paleta de placa plana

requerida para girar la cabeza hacia el viento. En 1918, una veleta aerodinámica con ocho veces el esfuerzo de torsión de la placa plana superó este problema.

Los anemómetros estáticos de tubo de Pitot modernos usan el mismo principio que en el anemómetro Dines, pero con un diseño diferente. La implementación utiliza un tubo pitot estático que es un tubo pitot con dos puertos, pitot y estático, que normalmente se usa para medir la velocidad del aire de los aviones. El puerto de Pitot mide la presión dinámica de la boca abierta de un tubo con la cabeza puntiaguda mirando hacia el viento, y el puerto estático mide la presión estática de pequeños agujeros a lo largo del costado de ese tubo. El tubo de Pitot está conectado a una cola para que siempre haga que la cabeza del tubo mire al viento. Además, el tubo se calienta para evitar la formación de hielo de escarcha en el tubo. Hay dos líneas desde el tubo hasta los dispositivos para medir la diferencia de presión de las dos líneas, como se observa en la Figura 12. Los dispositivos de medición pueden ser manómetros, transductores de presión o registradores de gráficos analógicos.

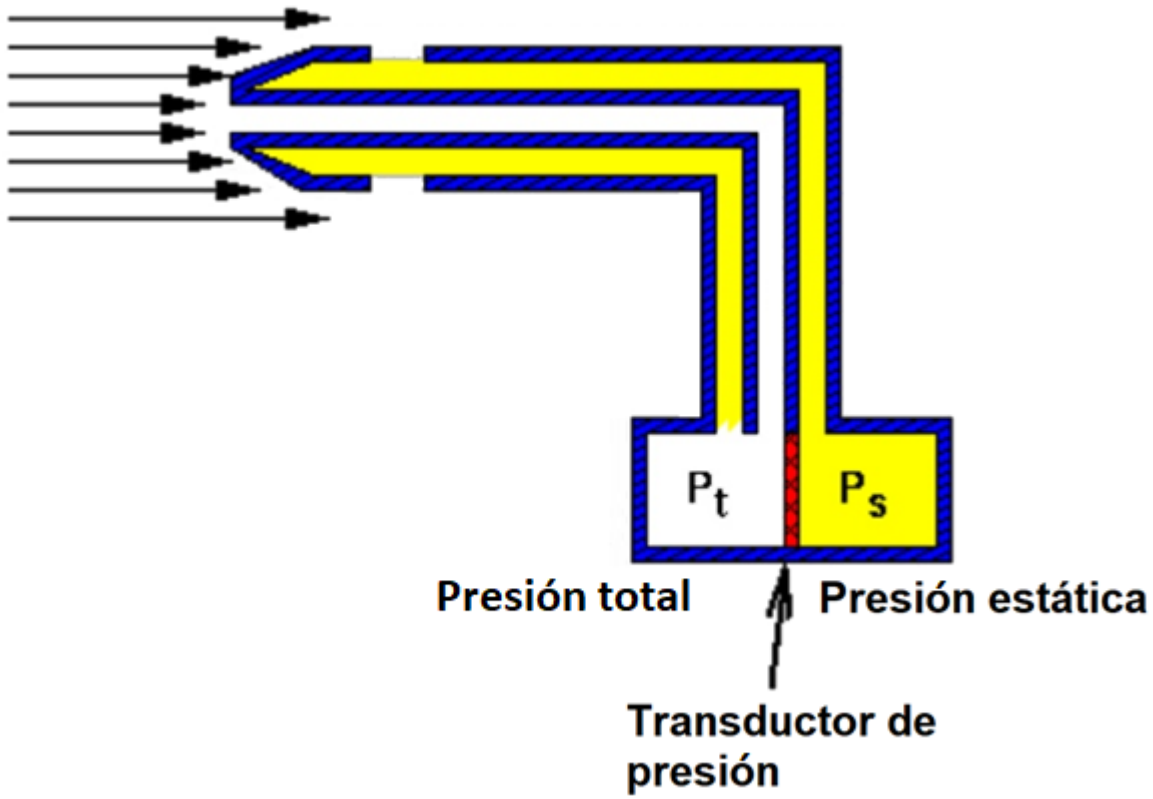

Figura 12 Tubo de pitot estático.

Redes neuronales artificiales

Las redes neuronales artificiales (RNA) son sistemas computacionales los cuales progresivamente mejoran su desempeño en tareas mostrándoles ejemplos sin programarles tareas específicas. Por ejemplo, en reconocimiento de imágenes, pueden “aprender” a identificar imágenes que contienen gatos analizando imágenes de muestra que han sido etiquetadas manualmente como “con gato” o “sin gato” y usar el resultado para identificar gatos en otras imágenes. Hacen esto sin ningún conocimiento previo de lo que define a los gatos, no saben si los gatos tienen pelaje, colas o bigotes, en vez de eso generan su propio conjunto de características del material de aprendizaje que procesan.

Componentes de una RNA

Una RNA está basada en una colección de unidades conectadas o nodos llamados neuronas artificiales. Cada conexión entre neuronas artificiales puede transmitir una señal de una a otra. Las neuronas artificiales que reciben la señal pueden procesarla y enviar otra señal a neuronas conectadas a esta.

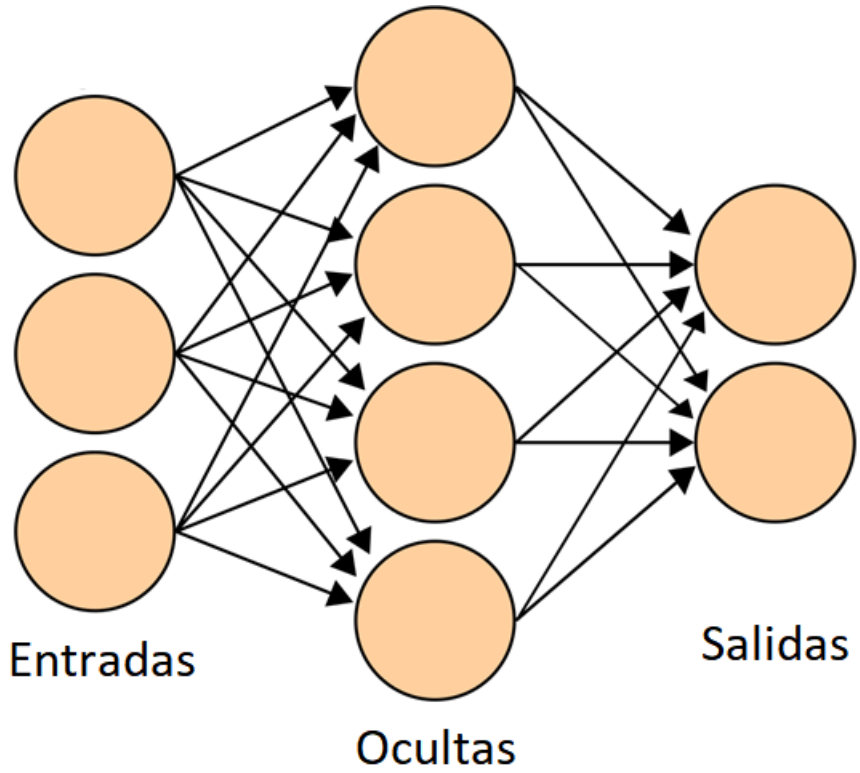

Figura 13 Representación clásica de una red neuronal artificial.

En una implementación común de redes neuronales artificiales, la señal en una conexión entre neuronas artificiales es un número real, y la salida de cada neurona

artificial es calculada por una función no lineal de la suma de las entradas. Las neuronas artificiales y sus conexiones típicamente tienen pesos que ajustan los procesos de aprendizaje. El peso incrementa o reduce la fuerza de la señal en una conexión. Las Neuronas artificiales pueden tener un umbral para solo enviar la señal cuando pase dicho umbral. Típicamente, las neuronas artificiales son organizadas en capas. Diferentes capas pueden realizar diferentes tipos de transformaciones a sus entradas. Las señales viajan de la primera capa (entrada) a la última capa (salida), pasando posiblemente por capas intermedias, como se observa en la Figura 13.

Neuronas

Una neurona j que recibe una entrada $p_j(t)$ de las neuronas predecesoras consta de los siguientes componentes

- Una activación $a_j(t)$, el estado de la neurona, dependiendo de un parámetro de tiempo discreto,
- Posiblemente un umbral θ_j, el cual se mantiene fijo a menos que lo cambie una función de aprendizaje,
- Una función de activación f que calcula la nueva activación en el instante t+1 a partir de $a_j(t)$, θ_j y la entrada neta $p_j(t)$ dando origen a la relación

  $$a_j(t+1) = f(a_j(t), p_j(t), \theta_j),$$

- Y una función de salida f_{out} que calcula la salida de la activación

  $$o_j(t) = f_{out}(a_j(t)).$$

A veces la función de salida es simplemente la función identidad.

Una neurona de entrada no tiene predecesoras, pero sirve como interfaz de entrada para la red. Similarmente una neurona de salida no tiene sucesoras y entonces sirve como interfaz de salida de toda la red.

Conexiones, pesos y bias

La red consiste en conexiones, cada conexión transfiere la salida de una neurona i a la entrada de una neurona j. De esta forma i es la predecesora de j y j es la sucesora de i. A cada conexión se le asigna un peso w_{ij} (Zeil 1994). Algunas veces se agrega un bias a la suma total de los pesos que sirve como umbral para mover la función de activación (Abbod 2007).

Función de propagación

La función de propagación calcula la entrada $P_j(t)$ para la neurona j a partir de las salidas $o_i(t)$ de las neuronas predecesoras y típicamente tiene la forma

$$P_j(t) = \sum_i o_i(t) w_{ij}$$

Cuando se agrega el bias a la función, la forma cambia a lo siguiente

$$P_j(t) = \sum_i o_i(t) \omega_{ij} + \omega_{0j}$$

Donde ω_{0j} es el bias.

Regla de aprendizaje

La regla de aprendizaje es una regla o un algoritmo que modifica los parámetros de la red neuronal, para que cada entrada dada a la red se produzca una salida favorable. Este proceso de aprendizaje típicamente modifica los pesos dentro de la red.

Modelos de una RNA

En la inteligencia artificial, una RNA hace referencia a modelos de redes neuronales; estos son modelos matemáticos esencialmente simples que definen una función f:X→Y. A veces, los modelos también están asociados con un algoritmo de aprendizaje en particular. Un uso común de la frase “modelo RNA” es en realidad la definición de una clase (donde los miembros de la clase se obtienen variando parámetros, los pesos de conexión, o específicos de la arquitectura, tales como el número de neuronas o su conectividad).

Función de red

La palabra red en el término “red neuronal artificial” se refiere a las interconexiones entre las neuronas en las diferentes capas de cada sistema. Un sistema ejemplar tiene tres capas. La primera capa tiene neuronas de entrada que envían datos a través de las sinapsis a la segunda capa de neuronas, y luego a través de más sinapsis a la tercera capa de neuronas de salida. Los sistemas más complejos tendrán más capas, algunos aumentando las de entrada y de salida de neuronas. Las sinapsis almacenan parámetros llamados “pesos” que manipulan los datos en los cálculos.

Un RNA se define típicamente por tres tipos de parámetros:

1. El patrón de interconexión entre las diferentes capas de neuronas

2. El proceso de aprendizaje para la actualización de los pesos de las interconexiones

3. La función de activación que convierte las entradas ponderadas de una neurona a su activación a la salida.

Matemáticamente, la función de red de una neurona $f(x)$ se define como una composición de otras funciones $g_i(x)$. Este se representa como una estructura de red, con flechas que representan las dependencias entre variables. Un tipo de composición ampliamente utilizado es la suma ponderada no lineal, donde $f(x) = K\left(\sum_i w_{\{i\}} g_{\{i\}}(x)\right)$, dónde K (denominado comúnmente como la función de activación) es una función predefinida, como la tangente hiperbólica o función sigmoide. La característica importante de la función de activación es que proporciona una transición suave como valores de entrada de cambio, es decir, un pequeño cambio en la entrada produce un pequeño cambio en la producción.

La Figura 14 representa una descomposición de tales f, con las dependencias entre las variables indicadas por las flechas. Estos pueden ser interpretados de dos maneras.

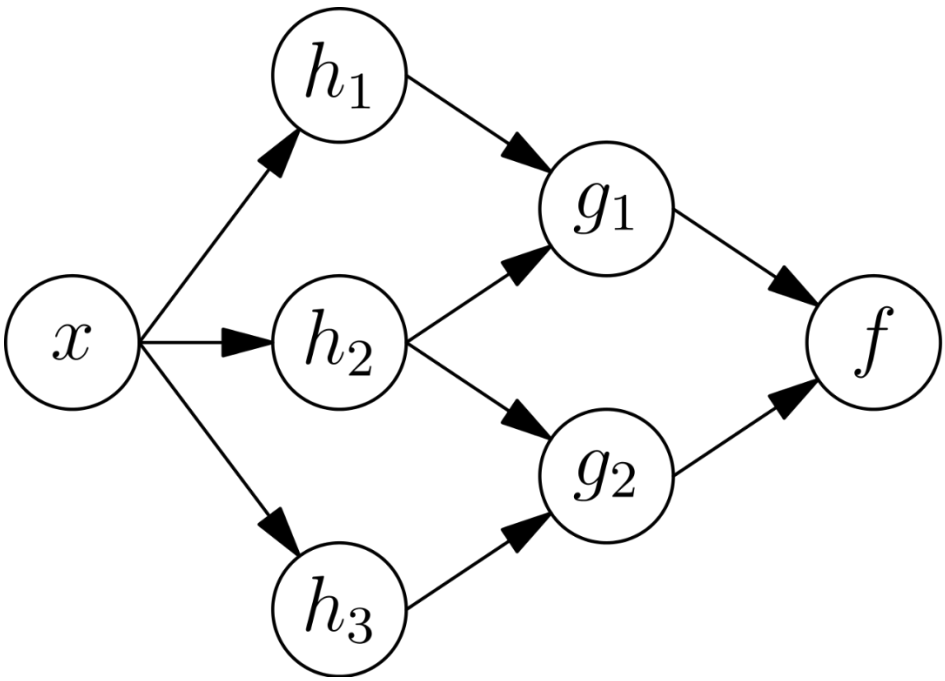

Figura 14. Dependencias de una RNA.

La primera, es la vista funcional: la entrada x se transforma en un vector de 3 dimensiones h, que se transforma a continuación en un vector de 2 dimensiones g, que es finalmente transformado en f. Este punto de vista se encuentra más comúnmente en el contexto de la optimización.

El segundo punto de vista es la vista probabilística: la variable aleatoria F = f (G) depende de la variable aleatoria G = g (H), Que depende de H = h (X), que depende de la variable aleatoria X. Este punto de vista se encuentra más comúnmente en el contexto de modelos gráficos.

Los dos puntos de vista son en gran medida equivalente. En cualquier caso, para esta arquitectura de red en particular, los componentes de las capas individuales son independientes entre sí (por ejemplo, los componentes de g son independientes entre sí, dada su aportación h. Esto permite, naturalmente, un grado de paralelismo en la ejecución.

Se dice que las redes como la anterior son de alimentación hacia delante, porque su gráfica es acíclica dirigida. Las redes con ciclos se denominan comúnmente recurrentes. Tales redes se representan comúnmente de la manera mostrada en la parte superior de la Figura 15, donde f se muestra como dependiente sobre sí misma. Sin embargo, no se muestra una dependencia temporal implícita.

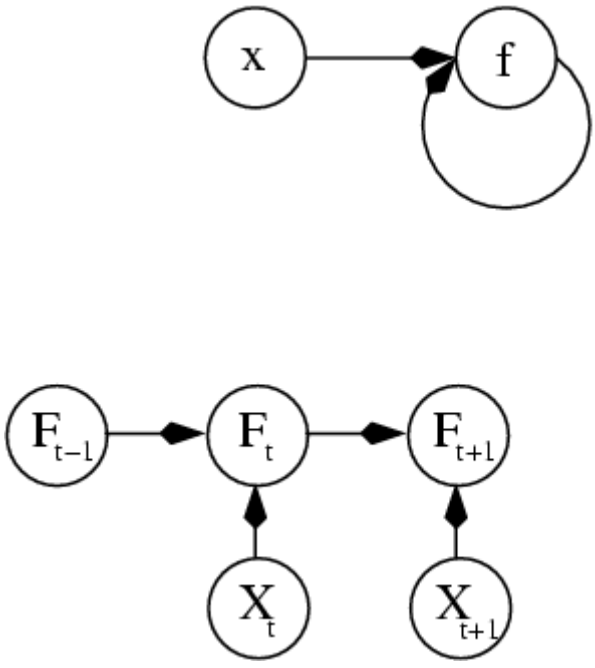

Figura 15 Dos representaciones separadas de una RNA recurrente.

El aprendizaje

Lo que ha atraído el mayor interés en las redes neuronales es la posibilidad de aprendizaje. Dada una determinada tarea a resolver, y una clase de funciones F, el aprendizaje consiste en utilizar un conjunto de observaciones para encontrar $f* \in F$ la cual resuelve la tarea de alguna forma óptima.

Esto implica la definición de una función de coste $C : F \rightarrow R$ tal que, para la solución óptima $f*$, $C(f*) \leq C(f) \forall f \in F$. Es decir, ninguna solución tiene un costo menor que el costo de la solución óptima.

La función de coste C es un concepto importante en el aprendizaje, ya que representa lo lejos que una solución particular se encuentra de la solución óptima. Los algoritmos de aprendizaje buscan a través del espacio de soluciones para encontrar una función que tiene el menor costo posible.

Para aplicaciones en las que la solución es dependiente de algunos datos, el costo debe ser necesariamente una función de las observaciones, de lo contrario no estaríamos modelando todo lo relacionado con los datos. Con frecuencia se define como una estadística a la que se pueden realizar sólo aproximaciones. Como un simple ejemplo, considere el problema de encontrar el modelo f, el cual minimiza $C = E[f(x) - y)^2]$, para pares de datos (x, y) extraídos de alguna distribución D. En situaciones prácticas sólo tendríamos N muestras de D y, por tanto, para el ejemplo anterior, solo tendríamos que minimizar $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Por lo tanto, el coste se reduce al mínimo a través de una muestra de los datos en lugar de toda la distribución de la generación de los datos.

Si bien es posible definir alguna función de coste, con frecuencia un coste particular, se utilizará, ya sea porque tiene propiedades deseables (tales como convexidad) o porque surge de forma natural a partir de una formulación particular del problema (por ejemplo, en una formulación probabilística la probabilidad posterior del modelo puede ser utilizada como un costo inverso). En última instancia, la función de coste dependerá de la tarea deseada.

Paradigmas de aprendizaje

Hay tres grandes paradigmas de aprendizaje, cada uno correspondiente a una tarea de aprendizaje abstracto en particular. Estos son el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo.

El aprendizaje supervisado

En el aprendizaje supervisado, se nos da una serie de ejemplos de pares (x, y), $x \in X$, $y \in y$ el objetivo es encontrar una función $f : X \rightarrow Y$ en las funciones de clase permitidas que correspondan con los ejemplos. En otras palabras, deseamos inferir el mapeo derivado de los datos; la función de coste está relacionado con la falta de coincidencia entre nuestro mapeo y los datos, y contiene implícitamente el conocimiento previo sobre el dominio del problema (Ojha et al. 2017).

Un coste de uso común es el error cuadrático medio, que trata de minimizar el error cuadrático medio entre las salidas de la red, f (x), y el valor objetivo y sobre todos los pares de ejemplos. Cuando uno trata de minimizar este coste utilizando descenso de gradiente para la clase de las redes neuronales llamadas perceptrones multicapas (MLP), se obtiene el común y bien conocido algoritmo de propagación hacia atrás para la formación de redes neuronales.

Tareas que caen dentro del paradigma de aprendizaje supervisado son el reconocimiento de patrones (también conocido como clasificación) y regresión (también conocido como aproximación de función). El paradigma de aprendizaje supervisado es aplicable también a los datos secuenciales (por ejemplo, reconocimiento del habla, del manuscrito, y de gestos). Esto se puede considerar como una forma de aprendizaje con un “maestro”, en la forma de una función que proporciona información continua sobre la calidad de las soluciones obtenidas hasta el momento.

Aprendizaje no supervisado

Para el aprendizaje no supervisado, se dan algunos datos x, la función de costo a minimizar, que puede ser cualquier función de x, de igual forma se da la salida de la red, f.

La función de coste depende de la tarea (lo que estamos tratando de modelar) y nuestras suposiciones implícitas (las propiedades de nuestro modelo, sus parámetros y las variables observadas).

Como un ejemplo trivial, considere el modelo f (x) = a donde a es una constante y el costo C = E [(x − f(x))^2]. Minimizar este coste nos dará un valor de a que es igual a la media de los datos. La función de coste puede ser mucho más complicada. Su forma depende de la aplicación: por ejemplo, en la compresión podría estar relacionada con la información mutua entre x y f (x), Mientras que, en el modelado estadístico, podría estar relacionada a la probabilidad posterior del modelo.

Las tareas que caen dentro del paradigma de aprendizaje no supervisado son en general problemas de estimación; las aplicaciones incluyen el análisis de grupos, la estimación de distribuciones estadísticas, la compresión de datos y el filtrado de señales.

Aprendizaje por refuerzo

En el aprendizaje por refuerzo, los datos x por lo general no se dan, pero son generados por la interacción de un agente con el medio ambiente. En cada instante en el tiempo t, el agente realiza una acción y_t y el medio ambiente genera una observación x_t y un costo instantáneo c_t, de acuerdo con algunas dinámicas. El objetivo es descubrir una política para la selección de las acciones que minimicen en cierta medida una función de un costo a largo plazo, por ejemplo, el costo acumulado esperado. La dinámica del medio ambiente y el costo a largo plazo para cada política general son desconocidos, pero pueden ser estimados.

Más formalmente el medio ambiente se modela como un proceso de decisión de Markov (MDP) con los estados s_1, …, $s_n \in S$ y las acciones a_1, …, $a_m \in A$ con las siguientes distribuciones de probabilidad: la distribución de costos instantáneos P(c_t | s_t), La distribución de observación P(x_t | s_t) y la transición P(s_{t+1} | s_t, a_t), mientras que una política se define como la distribución condicional sobre las acciones dadas las observaciones. Tomados en conjunto, las dos definen una cadena de Márkov (CM). El objetivo es descubrir la política (es decir, la CM) que minimice el costo.

Las RNAs se utilizan con frecuencia en el aprendizaje de refuerzo como parte del algoritmo general (Dominic et al 1991; Hoskins y Himmelblau 1992). La programación dinámica se ha unido a las RNA (dando lugar a la programación neurodinámica (Bertsekas y Tsitsiklis 1996) y se aplicó a problemas no lineales multidimensionales, como los implicados en enrutamiento de vehículos (Secomandi 2000), gestión de los recursos naturales (de Rigo et al. 2001; Damas et al. 2000) o

la medicina (Deng y Ferris 2008) debido a la capacidad de las RNAs para mitigar las pérdidas de precisión incluso cuando se reduce la densidad de la red de desratización para aproximar numéricamente la solución de los problemas de control originales. Las tareas que caen dentro del paradigma de aprendizaje por refuerzo son problemas de control, juegos y otras tareas de toma de decisiones secuenciales.

Tipo de entrada

Finalmente, también se pueden clasificar las RNAs según sean capaces de procesar información de distinto tipo en:

- Redes analógicas: procesan datos de entrada con valores continuos y, habitualmente, acotados. Ejemplos de este tipo de redes son: Hopfield, Kohonen y las redes de aprendizaje competitivo.
- Redes discretas: procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos booleanos. Ejemplos de este segundo tipo de redes son: las máquinas de Boltzmann y Cauchy, y la red discreta de Hopfield.

Algoritmos de aprendizaje

El entrenamiento de un modelo de red neuronal en esencia significa seleccionar un modelo de la serie de modelos permitidos (o, en un marco bayesiano, la determinación de una distribución en el conjunto de modelos permitidos) que minimiza el criterio de costo. Hay numerosos algoritmos disponibles para la formación de los modelos de redes neuronales; la mayoría de ellos puede ser vista como una aplicación directa de la teoría de optimización y la estimación estadística.

La mayoría de los algoritmos utilizados en las redes neuronales artificiales de formación emplean alguna forma de descenso de gradiente, utilizando propagación hacia atrás para calcular los gradientes reales. Esto se hace simplemente tomando la derivada de la función de coste con respecto a los parámetros de la red y a continuación, cambiando los parámetros en una dirección relacionada al gradiente. Los algoritmos de entrenamiento de propagación hacia atrás generalmente se clasifican en tres categorías:

- Gradiente descendiente (con tasa de aprendizaje y momento variables);

- Cuasi-Newton (Broyden-Fletcher-Goldfarb-Shannon, Método de la secante);
- Levenberg-Marquardt y gradiente conjugado (actualización Fletcher-Reeves, actualización Polak-Ribiere, Powell-Beale reinicio, gradiente conjugado escalado) (Forouzanfar et al. 2010).

Los métodos evolutivos (de Rigo et al. 2005), la programación de expresión génica (Ferreira 2006), el algoritmo de recocido simulado (Da y Xiurun 2005), el algoritmo esperanza-maximización, los métodos no paramétricos y la optimización por enjambre de partículas (Wu y Chen 2009) son algunos otros métodos para entrenar redes neuronales.

Algoritmo recursivo convergente de aprendizaje

Este es un método de aprendizaje específicamente diseñado para redes neuronales controladores de articulación (CMAC por sus siglas en inglés) de modelo cerebral. En 2004, un algoritmo recursivo de mínimos cuadrados fue introducido para entrenar redes neuronales CMAC (Ting et al. 2004). Este algoritmo puede convergir en un solo paso, y actualizar todos los pesos en un solo paso con cualquier dato nuevo de entrada. Al principio, este algoritmo tenía complejidad computacional de $O(N^{sa})$. Basado en factorización QR, este algoritmo recursivo de aprendizaje había sido simplificado para hacerlo O(N) (Ting et al. 2005).

El empleo de redes neuronales artificiales

Tal vez la mayor ventaja de las RNA es su capacidad de ser utilizadas como un mecanismo de función de aproximación arbitraria que “aprende” a partir de datos observados. Sin embargo, su uso no es tan sencillo, y una relativamente buena comprensión de la teoría subyacente es esencial. La elección del modelo dependerá de la representación de los datos y la aplicación. Modelos excesivamente complejos tienden a conducir a problemas en el aprendizaje.

Existen numerosas soluciones de compromiso entre los algoritmos de aprendizaje. Casi cualquier algoritmo va a funcionar bien con los hiperparámetros correctos para la formación de un conjunto específico de datos fijos. Sin embargo, la selección y el ajuste de un algoritmo para la formación en datos no previstos requieren una cantidad significativa de experimentación.

Si se seleccionan apropiadamente el modelo, la función de coste y el algoritmo de aprendizaje, la RNA resultante puede ser extremadamente robusta. Con la

aplicación correcta, las RNA pueden ser utilizadas de forma natural en el aprendizaje y aplicaciones de grandes conjuntos de datos. Su aplicación sencilla y la existencia de dependencias permiten implementaciones rápidas y paralelas en el hardware.

Las RNAs fueron desarrolladas originalmente como un medio para incorporar los dos aspectos esenciales del procesamiento de información humano: lógica e intuición (Teh 1995; Tan et al. 1996). Sin embargo, al pasar el tiempo, la atención se centró en resolver tareas específicas, desviándola de la biología. Las RNAs han sido usadas en una variedad de tareas, incluyendo visión por computadora, reconocimiento de lenguajes, traducción, filtros en redes sociales, jugar juegos de mesa y videojuegos, diagnósticos médicos y pronósticos.

Pronósticos

Técnicas de pronósticos

Pronosticar es el proceso de hacer predicciones del futuro basándose en datos pasados y presentes. Un ejemplo común es la estimación de alguna variable de interés en una fecha específica futura.

El riesgo y la incertidumbre son básicos en los pronósticos. Se considera como buena práctica indicar el grado de incertidumbre asociado al pronóstico. Se recomienda que los datos estén actualizados para que el pronóstico sea lo más exacto posible. En algunos casos los datos usados para predecir la variable independiente son pronosticados (French 2017).

Ha sido demostrado (Milligan et al. 1995; Watson et al. 1992) que el pronóstico de la potencia del viento es benéfico para la operación óptima de un sistema de potencia. Las aproximaciones que se encuentran en la literatura incluyen predicciones numéricas del clima y modelos de meso escala (Watson et al. 1993), Modelos de Markov equivalentes Generalizados (Kantz et al. 2004) y aproximaciones de series de tiempo. Este último incorpora varias técnicas como modelos ARMA, por sus siglas en inglés “autoregressive-moving-average” (Balouktsis et al. 1986; Kamal et al. 1997), filtros de Kalman (Bossanyi 1985), modelos autoregresivos de umbral liso y bilineal (ELSAM 1993) y, más recientemente, han sido propuestas técnicas de inteligencia artificial, las cuales incluyen el uso de perceptores multicapa (Lin et al. 1996; Beyer et al. 1994)

(Alexiadis et al. 1998), funciones radiales de base (Beyer et al. 1994), redes neuronales recurrentes (Barbounis et al. 2006), Lógica difusa (Kariniotakis et al. 1996) y una combinación de clasificador difuso con una red neuronal temporal (Wu et al. 1995). En general los modelos basados en inteligencia artificial superan a los modelos lineales y no lineales (Sfetsos 1999).

Exactitud

El error del pronóstico, también conocido como residual, es la diferencia entre el valor actual y el valor de la predicción para un periodo correspondiente.

$$E_t = Y_t - F_t$$

Donde E es el error del pronóstico para un periodo t, Y es el valor actual, y F es el valor del pronóstico.

Un buen pronóstico debe tener residuales no correlacionados y tener un promedio de 0. Si hay correlación entre valores residuales, significa que aún hay información en los residuales que debería ser usada para computar el pronóstico. Si los residuales tienen un promedio diferente de 0, la predicción esta sesgada.

Medidas de error agregado

Errores escalados

El error de pronóstico, E, está en la misma escala que los datos, por lo tanto, estas medidas no pueden ser usadas para comparar series de escalas diferentes dentro de estos errores se encuentran el error absoluto, la desviación absoluta promedio, el error cuadrático promedio, la raíz de error cuadrático promedio y el promedio de errores.

Tabla 1 Errores escalados.

Nombre	**Cálculo**
Error absoluto promedio (MAE por sus siglas en inglés)	$MAE = \frac{\sum_{t=1}^{N} \lvert E_t \rvert}{N}$ Donde N es el número de datos
Desviación absoluta promedio (MAD por sus siglas en inglés)	$MAD = \frac{1}{N}\sum_{i=1}^{N} \lvert x_i - m(X) \rvert$ Donde m(x) es el promedio.
Error cuadrático promedio (MSE por sus siglas en inglés)	$MSE = \frac{\sum_{t=1}^{N} E_t^2}{N}$

Raíz de Error cuadrático promedio (RMSE por sus siglas en inglés)	$RMSE = \sqrt{\frac{\sum_{t=1}^{N} E_t^2}{N}}$
Promedio de errores (E)	$\bar{E} = \frac{\sum_{i=1}^{N} E_i}{N}$

Errores porcentuales

Estos errores son usados más frecuentemente para comparar desempeño de pronósticos entre diferentes conjuntos de datos porque son de escala independiente. Sin embargo, tienen la desventaja de ser infinitos o indefinidos cuando Y es cercana o igual a cero. Estos tipos de errores incluyen el error porcentual absoluto promedio y la desviación porcentual absoluta promedio.

Tabla 2 Errores porcentuales.

Nombre	**Cálculo**
Error porcentual absoluto promedio (MAPE por sus siglas en inglés)	$MAPE = 100\frac{\sum_{t=1}^{N}\left\lvert\frac{E_t}{Y_t}\right\rvert}{N}$
Desviación porcentual absoluta promedio (MAPD por sus siglas en inglés)	$MAPD = \frac{\sum_{t=1}^{N}\lvert E_t\rvert}{\sum_{t=1}^{N}\lvert Y_t\rvert}$

Errores de tendencia

Se ha propuesto el uso de errores escalados en vez de los porcentuales, ya que tiene la ventaja de siempre estar definidos y ser finitos, excepto en el caso donde todos los datos sean iguales, pero es un caso irrelevante. Este tipo de error es fácilmente interpretable, dentro de esta categoría se encuentra el error escalado absoluto promedio.

Tabla 3 Errores en tendencia para pronósticos.

Nombre	**Cálculo**
Error escalado absoluto promedio (MASE por sus siglas en inglés)	$MASE = \frac{\sum_{t=1}^{N}\left\lvert\frac{E_t}{\frac{1}{N-m}\sum_{t=m+1}^{N}\lvert Y_t - Y_{t-m}\rvert}\right\rvert}{N}$ m=periodo del ciclo o 1 si no tiene ciclos

Por supuesto, aún hay situaciones donde se prefiera el uso de otros errores. Por ejemplo, si todas las series tienen la misma escala, entonces puede preferirse MAE ya que es más simple de explicar. Si todos los datos son positivos y mucho mayores a cero, MAPE puede preferirse por simplicidad. Sin embargo, en situaciones donde se usan escalas diferentes que son cercanas a cero o negativas, se sugiere el uso de MASE (Hyndman y Koehler 2006).

Cuando se compara la exactitud de diferentes métodos de pronóstico en un conjunto de datos específicos, se comparan las medidas de error agregado con cada una y se prefiere el método que produce el error más bajo.

Al evaluar el pronóstico es importante compararlo con datos que no fueron utilizados en el entrenamiento. Es común utilizar una porción de los datos disponibles para el entrenamiento y el resto para la prueba del modelo.

Limitaciones de los pronósticos

Hay muchos eventos y valores que no pueden ser pronosticados confiablemente. Eventos como el lanzamiento de un dado o los números de la lotería no pueden ser pronosticados porque son eventos aleatorios y no muestran relación entre los datos. Cuando los factores que rigen lo que se está pronosticando no se conocen o se comprenden bien, como en la bolsa de valores, los pronósticos son a menudo inexactos o incorrectos (Hyndman y Koehler 2006).

Redes neuronales sin computadora

El éxito de las redes neuronales en algunas áreas las ha conducido a ser aplicaciones comerciales que puedan trabajar sin necesitar una computadora (Dias et al. 2004).

La implementación más común de una RNA es en una computadora, la necesidad de dejar esa implementación puede surgir por muchas razones: reducir el costo de implementación, Lograr mayor velocidad de procesamiento o lograr implementaciones más simples. Esto puede obtenerse remplazando la computadora por un hardware especifico.

A diferencia de la arquitectura convencional de computadoras von-Neumann que es secuencial por naturaleza, las RNA toma ventaja del procesamiento paralelo

masivo (Liao, Y). Esto puede ser explotado por algún hardware para incrementar la velocidad de procesamiento.

Especificaciones del hardware

Desde el punto de vista del usuario, el primer paso para seleccionar el hardware adecuado es especificar las características de la RNA a implementar. Estas características incluyen el tipo de RNA (multicapa, función de base radial, Kohonen, etc.), el número de neuronas, número de entradas y salidas, el número de conexiones de cada neurona, la precisión, la velocidad de operación o desempeño y otras características que podrían ser más o menos importantes dependiendo de la aplicación.

La precisión usada debe ser un parámetro importante por considerar, ya que podría haber diferentes precisiones para varias partes: entradas, salidas, pesos, cálculos internos, acumuladores y multiplicadores.

Desempeño de las RNA sin computadora

La tasa de desempeño más común es conexiones por segundo (CPS por sus siglas en inglés) (Lindsey y Lindblad, 1994), definido como la tasa el número de multiplicación y acumulación de operaciones durante la fase de ejecución. Durante la fase de aprendizaje existe un equivalente, conexiones actualizadas por segundo (CUPS, por sus siglas en inglés) y mide cuantos pesos cambian por segundo.

También existen otras evaluaciones. El valor de CPS puede ser normalizado dividiéndolo por el número de pesos N_w, obteniendo las conexiones por segundo por peso (CPSPW, por sus siglas en inglés), el cual fue sugerido como una mejor forma de evaluación de desempeño (Holler, 1991). Otra evaluación puede ser conexiones primitivas por segundo (CPPS, por sus siglas en inglés), calculado como

$$CPPS = b_{in} \times b_w \times CPS \quad (1)$$

Donde b_{in} es el número de bits usados por las entradas y b_w es el número de bits usados por los pesos, esta evaluación permite que la precisión sea incluida en la evaluación de desempeño (Keulan et al., 1994; Schüffny et al., 1999). Estas dos evaluaciones son aplicables también a CUPS.

Otro parámetro que puede ser usado para analizar el desempeño es la disipación de potencia (Schüffny et al., 1999). Dependiendo de la tecnología, frecuencia del reloj, número de elementos de procesamiento, exactitud, etc., cada solución de hardware tiene una medida de disipación de potencia, que no puede ser comparada directamente. Evaluar la energía utilizada por conexión fue propuesto (Schüffny et al., 1999), sin embargo, esta evaluación sería un indicador de eficiencia energética en el circuito, además de que esta evaluación no puede ser realizada en la mayoría de los dispositivos.

Tipos de implementación de una RNA

Se propone una clasificación (Aybay et al., 1996) que divide al hardware en neuro-chips y neurocomputadoras, ambas con las subcategorías de propósito general y especial. Otra clasificación (Caviglia D) los divide en neurocomputadoras, aceleradores de computadoras personales, chips, librerías de células y microcomputadoras embebidas.

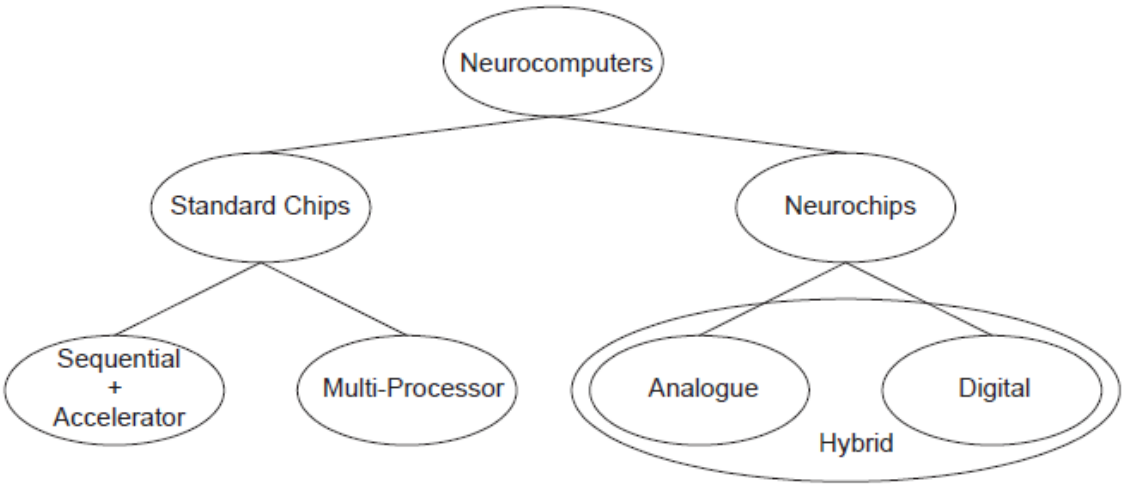

Figura 16 Categorías de hardware para redes neuronales según Heemskerk, 1995.

Otra clasificación (Heemskerk ,1995) es mostrada en la Figura 16. Aquí, todas las soluciones en hardware son llamadas neurocomputadoras, pueden estar echas de chips estándar o neurochips. Los chips estándar pueden ser clasificados como multiprocesos o como secuencial y acelerador. Los neurochips pueden ser clasificados como análogos, digitales o híbridos.

Implementaciones de las RNA sin computadora

Las implementaciones analógicas tienen la ventaja de obtener alta velocidad y densidad, pero tienen varias desventajas tecnológicas: almacenamiento de los pesos, estabilidad con la variación de la temperatura y obtención de multiplicadores lineales de rango amplio. Estos problemas resultan en una pérdida de precisión cuando se compara con otros tipos de implementaciones.

Las implementaciones digitales tienen la ventaja de guardar los pesos en memoria, son fácilmente integrados con otras aplicaciones, es más fácil implementar algoritmos de aprendizaje y definir el número de bits de los operandos y acumuladores. Comparados con los analógicos, los digitales suelen ser más precisos.

Las implementaciones digitales suelen dividirse en cuatro clases, arquitecturas de rebanadas, instrucción simple datos múltiples (SIMD, por sus siglas en inglés), dispositivos de arreglos sistólicos (Baratta et al., 1998) (Lindsey & Lindblad, 1994) y RBF, aunque se asume otras clases como chips multiprocesador y otras resultantes de la tecnología como PLD y FPGA.

Procesadores digitales de señales

Los procesadores digitales de señales (DSP por sus siglas en inglés) realizan operaciones matemáticas como señales digitales, los DSPs pueden realizar operaciones lineales y no lineales. Las operaciones de señales no lineales se relacionan con sistemas de identificación no linear (Billings et al, 2013), los sistemas DSPs pueden realizar tareas complejas de forma económica, como sintetizar y reconocer voz.

La aplicación de la computación digital al procesamiento de señales permite tener muchas ventajas sobre el procesamiento analógico, como la detección y corrección de errores en transmisiones y en compresiones (Broesch et al, 2008).

La ventaja clave de un sistema DSP sobre los basados en circuitos analógicos es que gracias a los avances rápidos en la manufactura de CI se ha incrementado la densidad y la velocidad de los DSPs y se han reducido los tamaños y los costos. Como resultado, los DSPs pueden combinar funciones de control digital e interconectarse con una amplia selección de dispositivos.

La flexibilidad es otra ventaja, a diferencia de los sistemas digitales que usualmente requieren modificación de hardware para tener un efecto en su funcionalidad. Los DSP pueden ser rápidamente reprogramados al cambiar los requerimientos.

Los sistemas DSP gozan de otras dos ventajas sobre los sistemas analógicos, los cuales, son más sensibles a condiciones medioambientales como la temperatura. La otra ventaja de los DSP es que estos siempre producirán la misma salida de las mismas entradas, en contraste con los sistemas analógicos que son precisos dentro de un rango, por lo tanto, dos fabricantes del mismo sistema podrían tener resultados significativamente diferentes. Estas diferencias hacen a los sistemas DSP predecibles, repetibles y adecuados para manufacturar en alto volumen. Los DSPs estuvieron alguna vez limitados a aplicaciones de gama alta. Pero ahora su costo beneficio los hace la mejor solución para productos de consumo.

Ya que la mayoría de los sistemas de procesamiento de señales realizan operaciones matemáticas complicadas en tiempo real, los DSPs están diseñados especialmente para acelerar la ejecución de cálculos numéricamente intensos y repetitivos. Los elementos comunes incluyen

- Circuitería para desempeñar múltiples operaciones acumulativas.
- Memoria de acceso múltiple, el cual le permite al procesador cargar múltiples operandos, como coeficientes de una red neuronal, de forma simultánea y en paralelo con una instrucción.
- Varios modos de direccionar memoria y características de control de flujo de programa diseñados para acelerar la ejecución de operaciones repetitivas.
- Periféricos especiales o interfaces de E/S que permiten al procesador interactuar eficientemente con otros componentes del sistema, como convertidores analógico-digital y memorias.

El enfoque es evaluar el desempeño de un procesador DSP. Muchos fabricantes han mejorado las habilidades de sus DSPs de propósito general añadiendo instrucciones y hardware de aceleración. Por ejemplo, Intel Corp. desarrollo extensiones MMX para procesadores Pentium e Integrated Device Technology agrego funcionalidades DSP básicas a su línea R4000 de procesadores RISC.

Desempeño de un DSP

El desempeño comúnmente es preocupante al momento de elegir un procesador DSP. Después de todo, la elección debe ser capaz de cumplir ciertas demandas en tiempo real. En contraste, las restricciones en el precio descartan el pago de desempeño innecesario. Estos requisitos opuestos conducen al diseñador a elegir un DSP que se desempeñe bien y que sea de bajo costo.

El desempeño de un DSP puede ser medido de muchas formas. La más común es el tiempo necesario para realizar cierta cantidad de procesos. Sin embargo, para ciertas aplicaciones puede ser más importante el consumo energético o el uso de la memoria. Una técnica ideal para medir el desempeño estudiaría el tiempo de ejecución, el uso de memoria y el consumo energético.

Enfoques tradicionales para la medición del desempeño usan métricas muy simples para describir el desempeño del procesador. La unidad de desempeño más común es MIPS (Million instructions per second), la cual es engañosa porque la cantidad de trabajo hecho por una instrucción puede variar de un procesador a otro. En otras palabras, las figuras MIPS son útiles solo para comparar una misma arquitectura.

MOPS (Million operations per second) sufre de un problema, lo que cuenta como una operación y cuantas son necesarias para realizar un trabajo varia de procesador en procesador.

Las MACS (Mulyiply accumulates per second) son otra técnica para medir el desempeño ya que son las operaciones de multiplicación y suma son básicas en un DSP como las multiplicaciones de matrices. La mayoría de DSPs pueden completar una MAC por ciclo de instrucción, lo que convierte a esta unidad en equivalente al MIPS. Además, las mediciones en MACS no toman en cuenta el ajuste de datos necesario para realizar una operación de multiplicación acumulación.

Ninguna MIPS, MOPS o MACS pueden medir desempeño secundario como uso de memoria y consumo energético. Esto es importante ya que, si se excede la capacidad de la memoria, el tiempo de ejecución se ve limitado. Adicionalmente, si es necesario utilizar una memoria externa más lenta, la velocidad del procesador se verá reducida. De igual forma, en una aplicación portable un procesador es inusable si su consumo energético sobrepasa la capacidad del a batería. Muchos

fabricantes informan de un consumo energético “típico” a una velocidad de reloj. Sin embargo, el consumo energético varía según las instrucciones y tipos de valores, por lo que dichas especificaciones no tienen la información suficiente.

Comparación de aplicaciones

Una aproximación común para evaluar sistemas computacionales es usar aplicaciones completas, o incluso varias aplicaciones. La evaluación de aplicaciones permite medir el uso de memoria y el consumo energético de un procesador. Y es más apropiado realizar la comparación entre diferentes familias de procesador.

Este enfoque es usado por la corporación de evaluación de desempeño estándar (SPEC, por sus siglas en inglés) en su evaluación de procesadores y sistemas para propósito general. En DSPs, algunos ejemplos incluyen codificadores de voz (CELP, VSELP, GSM, etc.), módems (V.34, V90, etc.), y programas de control de servomotores para discos rígidos.

En DSPs se pueden encontrar cuatro problemas con la evaluación de aplicaciones. Primero, la mayoría de las aplicaciones de los DSPs no están bien definidas para permitir comparaciones justas. Por ejemplo, dos implementaciones de un modem estándar pueden manejar aritmética con diferentes precisiones numéricas, dependiendo si su objetivo es lograr un menor error o minimizar la demanda al procesador. Segundo, con las aplicaciones más complejas, es imposible asegurar que el software sea optimo o casi óptimo. Entonces, la evaluación podría esta sesgada por el programador. Tercero. Las evaluaciones de aplicaciones completas tienden a medir el desempeño de un sistema no solo el del procesador. Aislar el desempeño de un procesador DSP de las evaluaciones o de otros componentes del sistema como memoria externa podría ser muy difícil. Por ultimo. Codificar una aplicación entera para múltiples procesadores podría tomar mucho tiempo de ingeniería, haciéndolo impráctico.

Debido a que las comparaciones que se van a realizar son en un mismo DSP, pero con software diferente y que el consumo energético no es un problema se sugiere tomar como medida de desempeño el tiempo de computo como se ve en Sepulveda, 2009.

Evolución diferencial

Propuesto por primera vez en 1997 (Storn, R., & Price, K. 1997), la Evolución Diferencial (ED) es un algoritmo simple pero efectivo para optimizaciones globales (Rönkkönen, J. 2009). Aunque fue diseñado para optimizar funciones de variables continuas y discretas, ED también ha sido aplicada con éxito a problemas combinatorios (Onwubolu, G. C., & Davendra, D. 2009).

ED pertenece a la clase de algoritmos evolutivos (AE), llamados así porque son métodos basados en población, mutación, recombinación y selección para evolucionar una colección de soluciones candidatas hacia un estado óptimo. Como la mayoría de los AEs, ED abusa de la población con la recombinación. Sin embargo, la ED no intenta imitar a la naturaleza, como las hormigas (Dorigo, M., et al 1996), abejas (Karaboga, D. 2005), o el sistema inmunológico (Dasgupta, D., & Attoh-Okine, N. 1997).

La evolución diferencial es un método de búsqueda paralela directa que utiliza NP vectores de parámetros de dimensión D, como población para cada generación G.

$$x_{i,G}, i = 1,2, \dots, NP$$

NP no cambia durante el proceso de optimización. Los vectores iniciales de población son escogidos aleatoriamente y deben cubrir por completo el espacio de parámetros. Como regla, se asumirá una distribución de probabilidad uniforme para todas las decisiones aleatorias a menos que se diga lo contrario. En caso de que una solución preliminar esté disponible, la población inicial puede ser generada añadiendo desviaciones aleatorias normalmente distribuidas a la solución nominal $x_{nom,0}$. ED genera nuevos vectores de parámetros añadiendo una diferencia ponderada entre dos vectores de la población a un tercer vector. Esta operación se conoce como mutación. Los parámetros del vector mutado son mezclados con los parámetros de otro vector predeterminado, el vector objetivo, para dar lugar al vector prueba. La mezcla de parámetros es comúnmente referenciada como cruce. Si el vector prueba produce un mejor resultado que el vector objetivo al ser evaluados en la función de costo, el vector prueba remplaza al vector objetivo en la siguiente generación. Esta última operación es llamada selección. Cada vector de la población tiene que ser una vez el vector objetivo.

Mutación

Para cada vector objetivo $x_{i,G}$,i=1, 2,…,NP, se genera un vector mutante de acorde a

$$v_{i,G+1} = x_{r1,G} + F(X_{r2,G} - x_{r3,G})$$

Con índices aleatorios r_1, r_2, r_3 ∈ {1,2,…,NP}, enteros, mutuamente diferentes y donde F >0. Los enteros escogidos aleatoriamente r_1, r_2 y r_3 son escogidos para que sean diferentes del índice en ejecución i, por lo tanto, NP debe ser mayor o igual a 4. F es un factor constante y real ∈ [0,2] el cual controla la amplificación de la variación diferencial ($x_{r2,G}$-$x_{r3,G}$). La Figura 17 muestra un ejemplo de dos dimensiones que ilustra los diferentes vectores que forman parte en la generación de $v_{i,G+1}$.

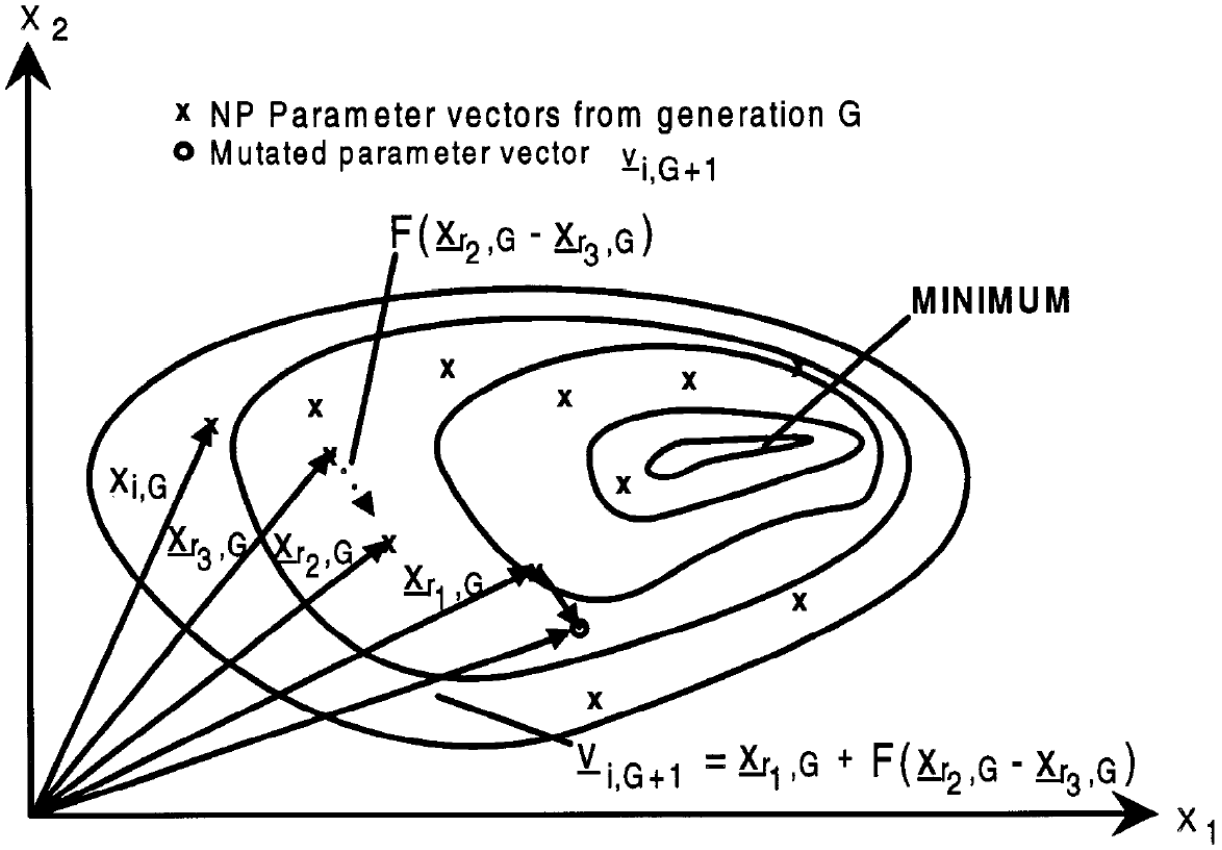

Figura 17 Ejemplo de una función costo de dos dimensiones, mostrando sus líneas de contorno y el proceso de generar $v_{i,G+1}$.

Cruce

Para lograr un incremento de diversidad en los vectores de parámetros, se introduce el cruce. Para este fin, el vector prueba es formado $u_{i,G+1} = u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1}$

donde

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & si(randb(j) \leq CR)\ o\ j = rnbr(i) \\ x_{ji,G} & si(randb(j) > CR)\ y\ j \neq rnbr(i) \end{cases}$$

$$j = 1,2, \dots, D.$$

En la ecuación anterior, randb(j) es la evaluación j-esima de un generador de números aleatorios uniformes con salida $\in$ [0,1]. CR es la constante de cruce que tiene que estar determinado por el usuario. rnbr(i) es un índice seleccionado aleatoriamente $\in$ 1,2,…,D, el cual asegura que $u_{i,G+1}$ tenga por lo menos un parámetro de $v_{i,g+1}$. La Figura 18 da un ejemplo del mecanismo de cruce para vectores de 7 dimensiones.

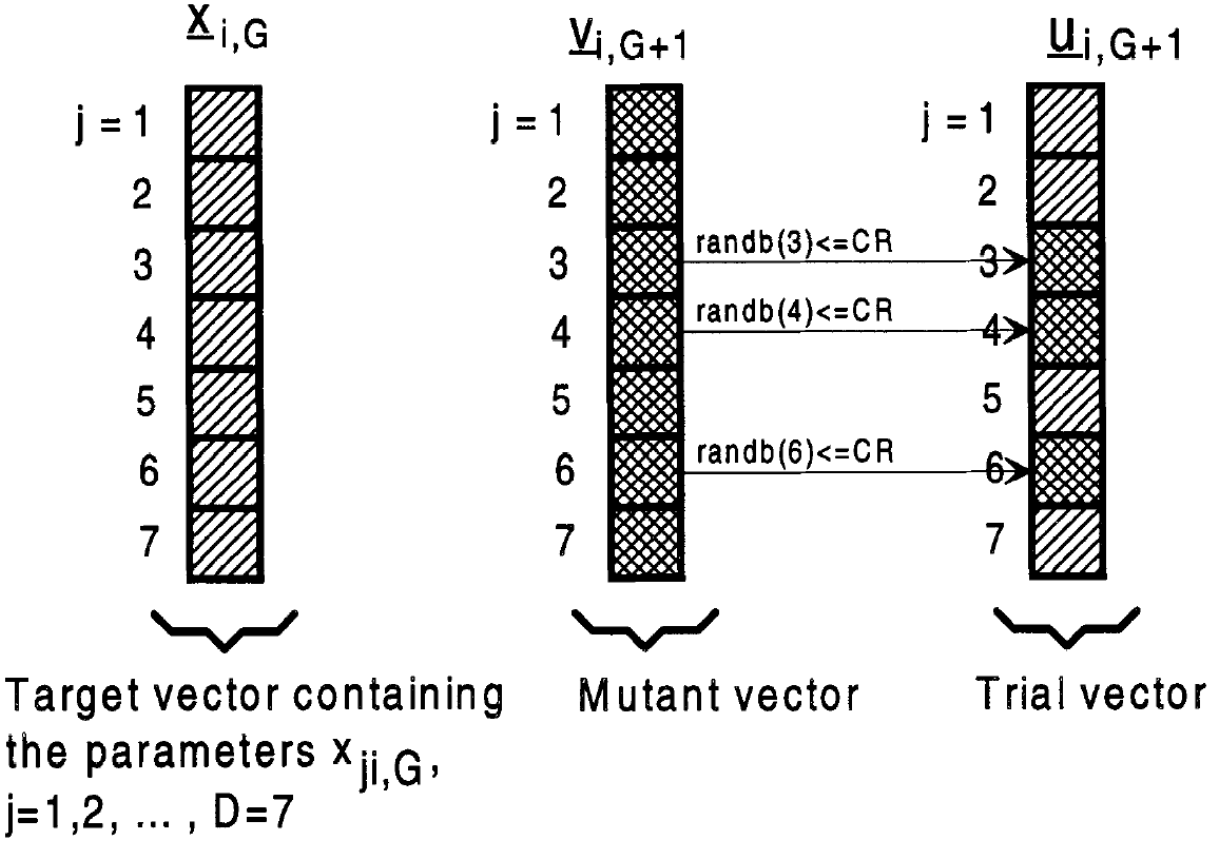

Figura 18 Se muestra el proceso de cruce con D=7 parámetros.

Selección

Para decidir si un vector debe o no convertirse en miembro de la generación G+1, el vector prueba $u_{i,G+1}$ es comparado con el vector objetivo $x_{i,G}$ usando una función de costo. Si el vector $u_{i,G+1}$ resulta en un valor más pequeño al evaluarlo en la función de costo entonces $u_{i,G+1}$ es establecido en $x_{i,G+1}$; de otra forma se mantiene el valor de $x_{i,G}$. La simplicidad de la ED es mejor explicada vía código de MATLAB, el cual, se encuentra en el Apéndice B.

Estado del arte

En la literatura, se estudiaron varias técnicas y se evaluó su desempeño en el contexto del problema de predicción de energía eólica (WPF). El objetivo no era construir un modelo WPF “completo” sino evaluar la capacidad de pronóstico de esas técnicas. En general, estas técnicas se utilizan para convertir las predicciones numéricas del clima (PNC) en energía eólica, el modelo denominado “viento a potencia (W2P)”.

Fugon et al. presentó un estudio sobre el rendimiento de diferentes modelos de minería de datos en WPF. Se examinaron dos versiones de regresión lineal: una es un modelo de regresión simple utilizado como referencia, y el otro consiste en combinar las variables de entrada para crear variables adicionales. Los modelos no lineales analizados fueron NN y SVM (Vapnik, 1999), el rendimiento de cada modelo se evaluó en tres parques eólicos ubicado en Francia por un horizonte temporal de 60 horas. Todos los modelos superaron la persistencia, y la superioridad global de los modelos no lineales se verificó en los tres parques eólicos. Sin embargo, el rendimiento de los modelos lineales es razonablemente bueno en comparación con el modelo de persistencia. El modelo no lineal con mejores resultados fue el modelo de bosques aleatorios. Los bosques aleatorios son una combinación de árboles predictores, donde cada árbol depende de los valores de un vector aleatorio independientemente muestreado con la misma distribución para todos los árboles en el bosque. El excelente rendimiento de este modelo significa que, como se encuentra en otros documentos mencionados anteriormente, el uso de varios modelos para WPF pueden disminuir el error de pronóstico.

Negnevitsky et al. En 2006 y 2007 abordaron el uso combinado de redes neuronales y lógica difusa en WPF. Este es un enfoque híbrido llamado Adaptive Neural Fuzzy System (ANFIS). Aunque este modelo solo se aplicó para pronósticos a muy corto plazo y los autores solo presentan resultados para este caso, se puede aplicar para horizontes de tiempo de entre 24 y 72 h.

Jursa en 2007, compara diferentes técnicas para pronósticos de energía eólica, como una red neuronal clásica MLP, mezcla de expertos (Weigand et al., 1995), SVM y la búsqueda del vecino más cercano con un algoritmo de optimización de enjambre de partículas (PSO) para la selección de los datos de entrada (Jursa, 2007). El autor combina adicionalmente diferentes modelos promediando las salidas del modelo. Se compararon los resultados de 10 parques eólicos ubicados en Alemania, y los PNC estaban disponibles para cada parque eólico. El mejor modelo fue el conjunto con tres modelos diferentes (es decir, una mezcla de expertos, vecinos más cercanos y SVM), con una mejora del 15% con respecto a una NN. Sin embargo, usar todos los modelos combinados no siempre es la mejor solución. De hecho, el conjunto con los cuatro modelos, en algunos parques eólicos, han registrado una mejora menor en comparación con el conjunto de tres modelos. El mejor modelo individual fue el modelo de mezcla de expertos, que logró una mejora del 8.8% sobre la NN. Los resultados de la SVM siempre son mejores cuando se comparan con los resultados de la red neuronal. El modelo de vecino más cercano era mejor que la NN en algunos parques eólicos. Sin embargo, en otros, la NN se comportó mejor. Los resultados mostraron ventajas de combinar varios modelos para pronósticos diarios. La intención con esta investigación era mejorar el modelo comercial Wind Power Management System (WPMS) desarrollado por el Institut für Solare Energieversorgungstechnik (ISET).

Kusiak et al. en 2009 probaron cinco modelos de minería de datos para producir pronósticos para un muy corto horizonte temporal (de 1 a 12 horas más adelante) y horizontes a corto / largo plazo (de 3 a 84 horas más adelante) usando PNC con el modelo del ciclo de actualización rápida (Rapid Update Cycle, RUC) y la mesoescala de América del Norte (NAM), respectivamente. Dos diferentes metodologías de pronóstico han sido comparadas y analizadas: (i) un método de pronóstico directo, en el que la generación de energía se pronostica directamente desde los datos de pronóstico del clima; y (ii) un método integrado, en el que se

pronostica la velocidad del viento con los datos del clima, y luego la potencia se pronostica con la velocidad del viento pronosticada con un algoritmo del vecino más cercano. Los autores utilizaron un algoritmo de boosting tree (Friendman, 2002) para seleccionar los datos más relevantes PNC que rodean el parque eólico. Por otra parte, incluso después de la selección, se aplica un análisis del componente principal (PCA) para reducir la dimensión de las entradas.

Los cinco modelos de extracción de datos fueron: SVM, MLP NN, RBF NN, árboles de regresión y bosques aleatorios. El MLP NN supera a los otros cuatro modelos tanto a corto plazo. El enfoque directo superó al enfoque integrado también para corto plazo. Los autores destacaron la fuerte dependencia entre la precisión del modelo de WPF y la precisión del modelo de PNC.

Duran et al. en 2007 estudió los modelos ARX para predicciones a corto y muy corto plazo con PNC y generación de datos en-línea como entradas. Notaron que la mejora obtenida en horizontes de tiempo corto (de 6 h) fue menor como resultado de la creciente relevancia de la potencia de salida pasada en comparación con el PNC para este horizonte de tiempo. Por otra parte, cuando el horizonte de tiempo es de 24 horas, la potencia de salida pasada pierde importancia, y el PNC gana más relevancia. Cuando se compara con el AR, la mejora del ARX es de 14% durante 24 horas y 26% en cuando se compara con la persistencia.

Barbounis y Theocharis en 2006 emplearon redes neuronales recurrentes locales para pronostica velocidad del viento y potencia 72 horas por adelante, con base en información meteorológica. Tres tipos de redes neurales recurrentes locales se estudiaron: (i) perceptron multicapa de respuesta al impulso infinito (IIR-MLP); (ii) la red multicapa de retroalimentación de activación local (LAF-MLN); y (iii) la red neuronal recurrente diagonal (RNN). Se presentan dos nuevos y eficientes algoritmos de aprendizajes: uno global y una aproximación de desacople del error de predicción recursivo (RPE) para entrenar la red neuronal en-línea (es decir, mediante la actualización de los pesos y el sesgo en línea). En el RPE global (GRPE), todos los pesos se actualizan simultáneamente. Por otra parte, para hacer frente a la mayor complejidad computacional del GRPE, una versión local del algoritmo, llamado RPE desacoplado (DRPE), se desarrolló. El DRPE consiste en dividir el problema de la optimización global en un conjunto de subproblemas

manejables a nivel de neurona. Al hacerlo, es posible reducir los requisitos de computación y almacenamiento considerablemente, mientras se preserva la alta precisión del GRPE. Las tres redes recurrentes se compararon con dos modelos estáticos, una respuesta de impulso finito NN (FIR-NN) y una red estática convencional MLP. El rendimiento de los modelos propuestos se probó en un parque eólico ubicado en la isla griega de Creta, y los PNC fueron proporcionados por el pronóstico regional SKIRON para cuatro puntos que estaban a 30 km del parque eólico. Los resultados mostraron que las redes recurrentes se desempeñaron mejor en comparación con los modelos estáticos en todos los tiempos de anticipación. El FIR-NN superó al MLP estático en un 11.82% y 12.7% en términos de error absoluto medio. Los modelos recurrentes también lograron una mejora promedio del 50% (para tiempos de anticipación de más de 20 horas) sobre persistencia. Resultados similares son válidos para el pronóstico de velocidad del viento. Debido a la riqueza de la arquitectura de red, IIR-MLP presentó el mejor rendimiento en comparación con los otro dos. Las principales contribuciones de este documento son las nuevas redes neuronales de capacitación en línea y sus algoritmos.

Estos nuevos algoritmos de entrenamiento brindan la capacidad de hacer frente a los cambios en el comportamiento y operación del parque eólico, así como un bajo esfuerzo computacional. Bessa et al. en 2009 informó el uso del algoritmo de back-propagation para entrenar directamente una red neuronal en línea. La metodología es de la siguiente manera: (i) entrenar una red neuronal utilizando un enfoque de propagación inversa por lotes con el datos históricos disponibles; (ii) en el modo en línea, la NN hace predicciones para la muestra de tiempo t + k en el momento t ; y (iii) cuando se conoce el valor medido, la red neuronal pronostica nuevamente para el tiempo t , y el error de pronóstico que tuvo lugar en t (en el nuevo valor medido) se calcula y propaga a través de la red (ponderaciones y sesgos son actualizados) solo una vez. Esta metodología permite tratar adecuadamente los flujos de datos en presencia de derivación de conceptos o cambios de concepto, que ocurren en el comportamiento del viento. Al mismo tiempo, estos cambios conceptuales también son el resultado de las operaciones prácticas de los parques eólicos, causado por variaciones en la capacidad de generación disponible ya sea por mantenimiento o falla o simplemente por adiciones de capacidad. Los resultados

de dos parques eólicos ubicados en Portugal han demostrado que hubo una mejora con el uso del entrenamiento de la red neuronal en línea, no solo en el funcionamiento normal, sino también en el caso de un cambio de concepto.

Es bien sabido que la velocidad del viento frente a la curva de potencia de una turbina eólica es altamente no lineal. La transformación de la velocidad del viento en energía eólica cambia las propiedades estadísticas de los errores. Este resultado se ha demostrado (Lange, 2005), para seis sitios en Alemania, donde las distribuciones de error de los modelos de predicción de la energía eólica estaban sesgadas a la derecha y tenían un positivo exceso de curtosis. Esto significa que eran asimétricos; presentaron una mayor frecuencia de error a la izquierda de la media y eran más planos que la distribución de Gauss. Los autores declararon el seguimiento:

“La desviación estándar relativa de la potencia de salida medida es más grande por un factor de 1.8-2.6 que la desviación estándar relativa, de las series de tiempo de la velocidad del viento medida. Este factor fue causado por la curva de potencia y puede considerarse como el factor efectivo de no linealidad que describe el escalamiento de las variaciones en la velocidad del viento debido a la pendiente local de la curva de potencia ... “. La misma forma de distribución de error se puede encontrar en otras publicaciones (Bludszuweit et al., 2008; Fabbri et al., 2005).

Al observar la literatura, es posible entender que, de una forma u otra, los modelos dependen de un proceso de capacitación y generalmente adoptan el Error Cuadrático Mínimo (MSE) como un criterio de calidad. La aplicabilidad de MSE para entrenar a un mapeador (cualquier modelo mapeando una relación entrada-salida, como redes neuronales, sistemas de inferencia difusa, series temporales u otras) solo es óptimo si la función de distribución de probabilidad (pdf) de los errores de predicción son gaussianos (Bishop, 1995). Minimizar el error cuadrado es equivalente a minimizar la varianza de la distribución de error. Usando este criterio, los momentos superiores (p. Ej., Sesgo, Kurtosis) no son capturados. Sin embargo, contienen información que debe transmitirse a los parámetros (pesos) de la red neuronal en lugar de permanecer en la distribución de error.

La presencia de distribuciones no gaussianas ha alentado nuevas investigaciones para nuevas técnicas para entrenar “mapeadores”. En WPF, tener un buen

“mapeador”, significa que es posible proporcionar mejores estimaciones al modelo W2P. Para lidiar con este problema, han sido desarrolladas funciones de costo alternativas.

Capítulo 3. Metodología

Una tarjeta de medición de potencia permite la estimación de esta a partir de las señales de corriente y de voltaje, es decir, con este dispositivo podemos reemplazar el uso de voltímetros y amperímetros, adicionando la capacidad de medir la energía producida e incorporando beneficios adicionales. Estos dispositivos implementan la comunicación por medio de protocolos ZigBee, con los que podemos integrar estos equipos a softwares para realizar gestión de energía a partir de las señales obtenidas de un sistema de monitoreo.

Un sistema de monitoreo de indicadores energéticos es capaz de registrar variables eléctricas de interés que ayudan a establecer el comportamiento de un sistema de potencia. Es importante contar con estos sistemas en donde se midan en tiempo real las variables de los equipos y sistemas a fin de vigilar su desempeño energético y operativo.

El dispositivo considerado en este trabajo como fuente de las señales a predecir por el algoritmo computacional es el diseñado en (de la Cruz May 2018). Esta tarjeta de medición cuenta con dos transductores de voltaje y cuatro transductores de corriente, los cuales convierten y reducen las variables energéticas de entrada en otras variables de salida para que pueden ser leídos por un microcontrolador después de ser acondicionadas con filtros. Se conecta un módulo Xbee para llevar a cabo una comunicación con el ordenador y obtener los datos de medición en tiempo real.

Los Xbee son dispositivos inalámbricos fabricados por Digi International, tienen su propio protocolo de comunicación por radio frecuencia, son robustos, de bajo costo, bajo consumo y tienen un alcance en sus distintos modelos entre 100 metros y los 10 kilómetros. Utilizan el protocolo IEEE 802.15.4 mejor conocido como ZigBee, estos fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible.

Para diseñar un circuito es necesario distribuir los componentes uniformemente en un software de diseño de PCBs, como se muestra en la Figura 19. Posteriormente se obtienen las coordenadas para su elaboración en una máquina CNC de circuitos impresos, y por último, se montan y sueldan los componentes, obteniendo como resultado una tarjeta como en la Figura 20.

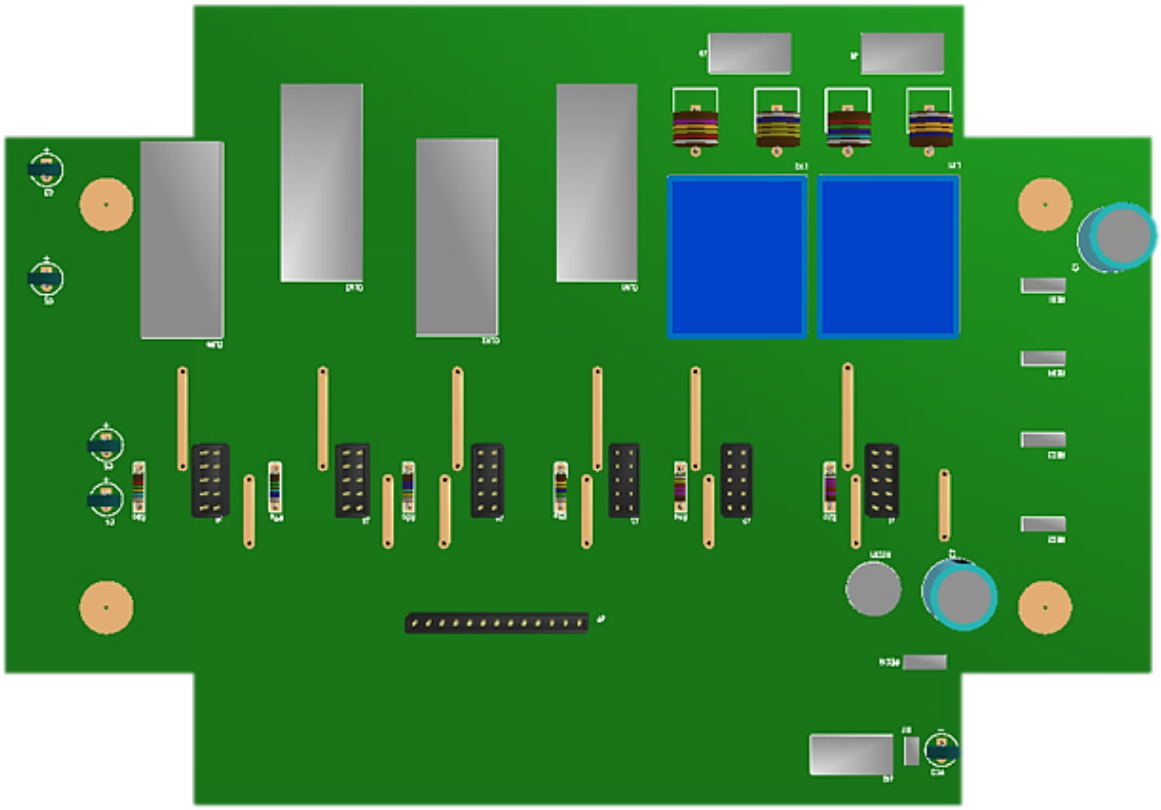

Figura 19 Diagrama de una tarjeta de medición de potencia (de la Cruz May 2018).

Figura 20 Tarjeta de medicion de potencia ya manufacturada (de la Cruz May 2018).

Existen sistemas para la medición de energía que ponen al alcance de los usuarios prestaciones de calidad de energía y análisis que antes eran exclusivos de los medidores de gama avanzada, como los medidores Fluke. Estos medidores permiten recopilar, almacenar y analizar información de los sistemas, ayudando a todo aquel usuario que busque conocimiento total de su consumo y que requiera alta confiabilidad y disponibilidad en la red eléctrica.

Los analizadores de calidad de potencia Fluke son una opción para implementar sistemas de gestión de energía. Estos permiten localizar, predecir, evitar y solucionar problemas en los sistemas de distribución de alimentación. La solución de problemas es rápida gracias a la visualización en pantalla de tendencias y sucesos capturados, mientras la grabación de datos continúa. Los sistemas Fluke cuentan con la herramienta de localización de averías completa: mide la tensión, la corriente, la frecuencia, la potencia, la fluctuación, la armonía, y el consumo de energía.

Las desventajas de emplear estos sistemas para la gestión energética como monitoreo de la producción de energía de fuentes renovables son el elevado costo de los equipos y el riesgo que implica dejar el equipo en cuartos de control por tiempo indefinido (si es que existe alguno cercano al área de interés), donde podría presentar alguna avería y ser sujeto a robo.

Otra de las desventajas que presentan los sistemas Fluke, es en la obtención de datos, ya que se requiere extraerlos de forma manual, donde una tarjeta con comunicación inalámbrica es ideal para el monitoreo en tiempo real de generación de energía. Con esta característica la tarjeta sólo necesita ser instalada en el lugar que se desee tomar mediciones, sin la necesidad de extraer manualmente los datos, además el costo de elaboración de la misma es mucho menor al de los equipos Fluke y otros sistemas.

Los datos de velocidad de viento y potencia generados se obtuvieron de los registros de mediciones de un parque eólico en Baja California Sur (Castro 2018). Los datos fueron tomados a una altura de 60 m entre febrero y agosto de 2015 cada 5 minutos. Posteriormente esos datos fueron promediados y convertidos a cada 15 minutos, con el fin de minimizar el tiempo de cómputo cuando se ejecuten los algoritmos de predicción. Los datos de potencia utilizados en el entrenamiento y

correspondientes al mes de febrero se muestran en la Figura 21, y en la Figura 22, se muestran los datos utilizados en la evaluación y corresponden al mes de marzo. El orden óptimo de retrasos del sistema fue calculado en 8 según la tesis previa a esta por el método de embedding de Cao (Castro 2018).

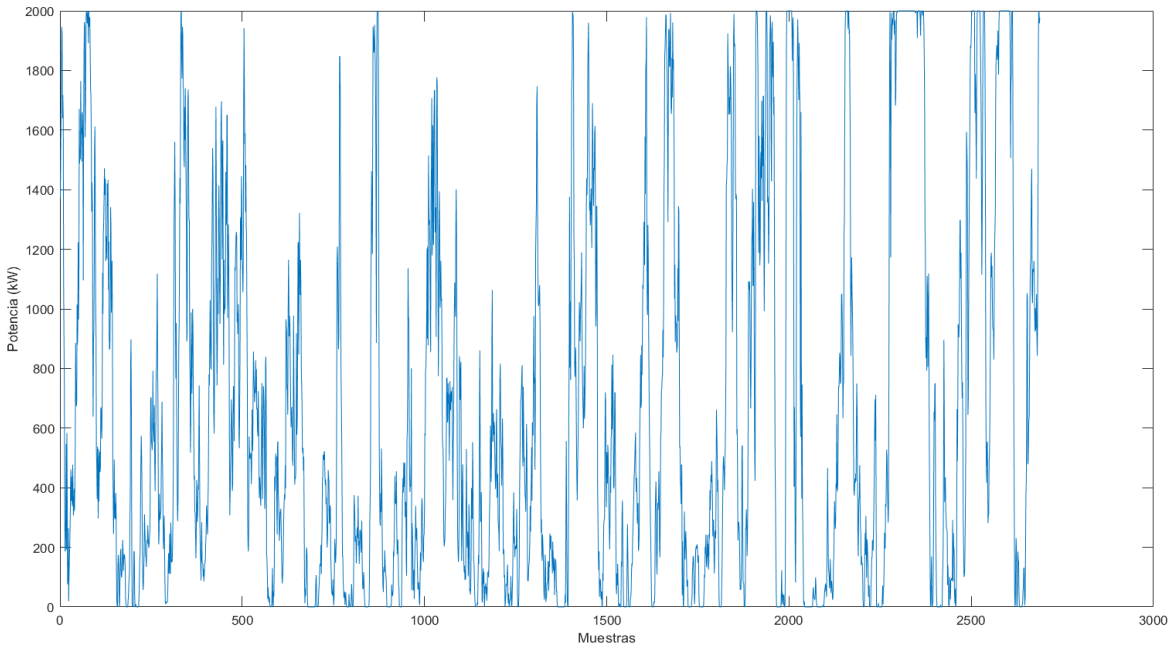

Figura 21 Historial de potencia utilizada para el entrenamiento.

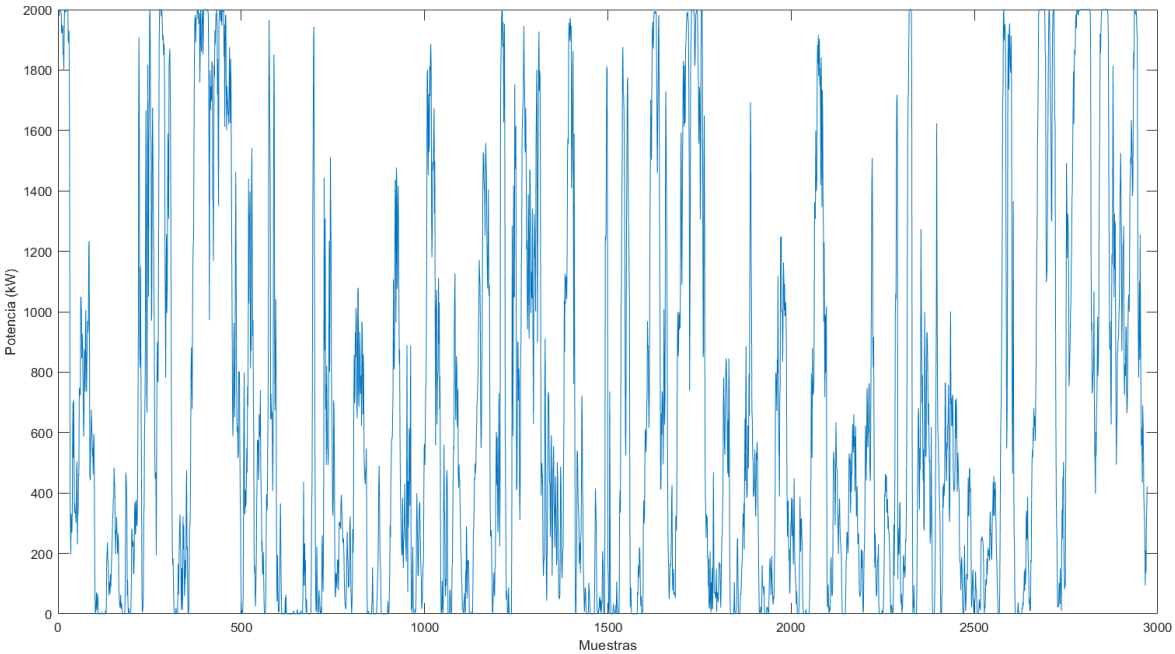

Figura 22 Historial de potencia utilizado para la evaluación.

Los parámetros que se utilizaron en la red neuronal son los mostrados en la Tabla 4 y la estructura se muestra en la Figura 23. De igual forma los parámetros del

algoritmo de evolución diferencial se encuentran en la Tabla 5, este conjunto de parámetros se conoce en la literatura como DE/rand/1/bin.

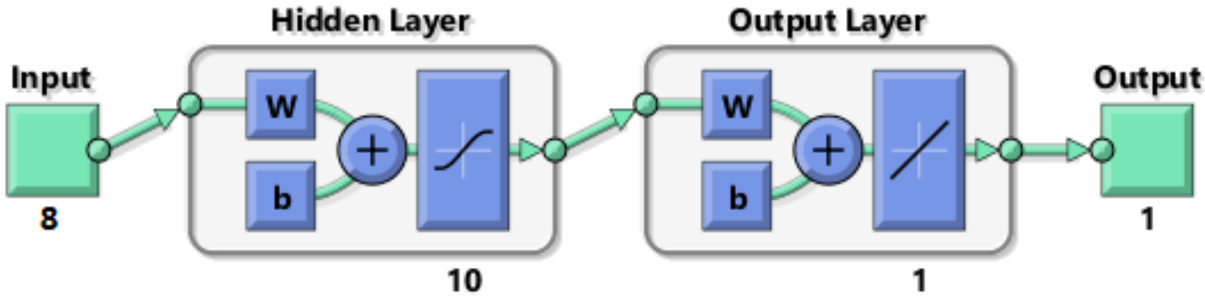

Figura 23 Estructura de la red neuronal utilizada.

Tabla 4 Parámetros utilizados en el modelo de la red neuronal.

Parámetro	**Valor**
No. de Neuronas Ocultas	10
No. de Elementos de retardo	8
No. de Capas Ocultas	1
Función de Transferencia capa Oculta	Sigmoide
Función de Transferencia capa de Salida	Lineal
Algoritmo de Entrenamiento	Filtro extendido de Kalman

Es relevante mencionar que el código utilizado en la tesis pasada estaba lejos de ser el óptimo para una red neuronal, tardaba más de 100 segundos en cada época, por lo que se recodificó, logrando un tiempo promedio por época de 1.25 segundos, lo que significa una mejora en tiempo de ejecución de alrededor del 9,000%. Este algoritmo puede ser consultado en el Apéndice A. Las diferencias de código consistían principalmente en que en el código pasado se utilizaban ciclos for para realizar las operaciones entre matrices, a diferencia de la implementación actual que utiliza operaciones directas entre matrices. Además, el código previo realizaba varias transformaciones a los datos en varias líneas o utilizando ciclos for anidados, lo que en este trabajo se optimizó utilizando propiedades matriciales. Por último, en

el código previo se introducían manualmente valores en las funciones lo cual fue automatizado en este trabajo, de tal forma que el código paso de 330 líneas a solamente 120. Sin embargo, el código actual aún puede recibir mejoras en tiempo de ejecución.

Tabla 5 Parámetros utilizados en el algoritmo de evolución diferencial.

Parámetro	**Valor**
Inicialización de población	Aleatorio
No. de vectores en la población	5
No. de vectores en la mutación	1
Esquema de cruce	bin
Cantidad de pesos	9
Probabilidad de cruce	.5
Peso de la mutación	.5

Para comparar el desempeño de la red neuronal se programó el algoritmo de evolución diferencial rand/1/bin, como se describió previamente. Como modelo de predicción se utiliza un modelo ARX el cual se muestra en la ecuación siguiente, y como función de costo el error calculado mediante RMSE entre la predicción y el valor real. El promedio en tiempo de ejecución por generación es de 1.16 segundos.

$$y_{k+1} = \sum_{i=1}^{n_a} w_i y_{k-i+1} + w_b$$

Donde:

y_{k+1} es la predicción,

w_i son los pesos,

y_{k-i+1} son los valores históricos,

w_b es el bias,

i es el retraso,

k es el momento actual, y

n_a es la cantidad de retrasos.

Implementación en hardware

Después de realizar los algoritmos, el siguiente paso es implementarlos en hardware para observar si se obtienen los mismos resultados, para eso se seleccionó el DSP F28379D LaunchPad de Texas Instrument por ser la mejor tarjeta de desarrollo que se tiene disponible en la facultad. En la Figura 24 se puede apreciar la tarjeta, algunas de sus características incluyen procesador dual de 200MHz, 1MB Flash, 204kB RAM y cuatro puertos UART.

Figura 24 Tarjeta F28379D LaunchPad de Texas Instruments.

Para ejecutar los algoritmos en la tarjeta se realizó un programa en simulink el cual se comunica con la tarjeta y se obtienen los resultados de vuelta en MATLAB. Un ejemplo del modelo utilizado se encuentra en el Apéndice C y en la Figura 25 se observa el diagrama de flujo de la implementación para la RNA. Estos modelos tienen que estar configurados para el hardware especifico y seleccionar que serán ejecutados por un hardware externo.

Se logró obtener los datos de velocidad de viento de un parque eólico, los cuales fueron utilizados para entrenar y evaluar los algoritmos de ED y la RNA, los cuales fueron implementados en Matlab. Posteriormente, se realizaron ajustes al código para poder implementarlo en hardware, específicamente en una tarjeta F28379D LaunchPad de Texas Instruments, la tarjeta fue configurada para comunicar los resultados a MATLAB, para lograrlo se tuvo que optimizar el manejo de memoria.

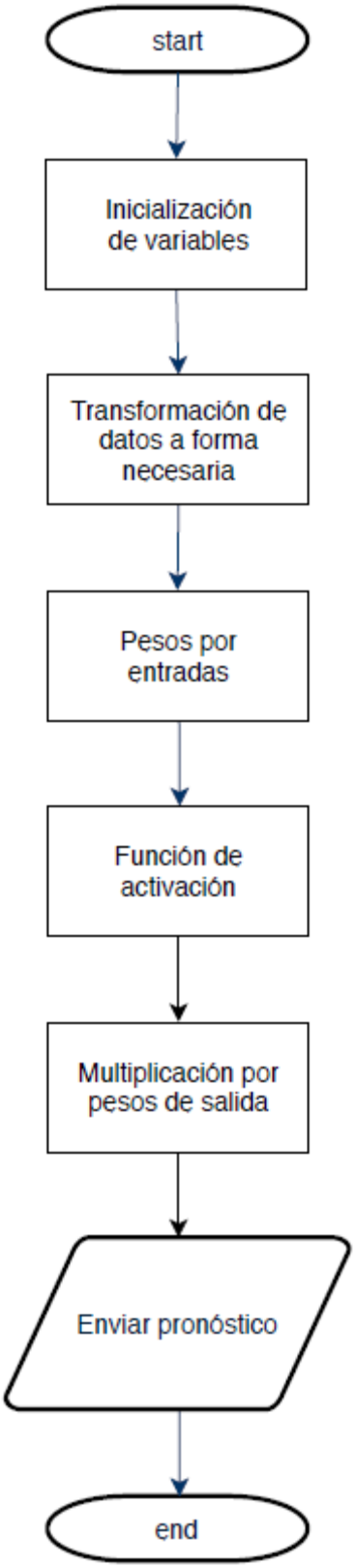

Figura 25 Diagrama de flujo de la RNA implementada en hardware.

Capítulo 4. Resultados

En este capítulo se presentan los resultados obtenidos en la fase experimental para la RNA y la ED, se evalúa el desempeño en hardware de cada algoritmo por separado, para posteriormente realizar una comparación entre ellos.

Predicción utilizando evolución diferencial

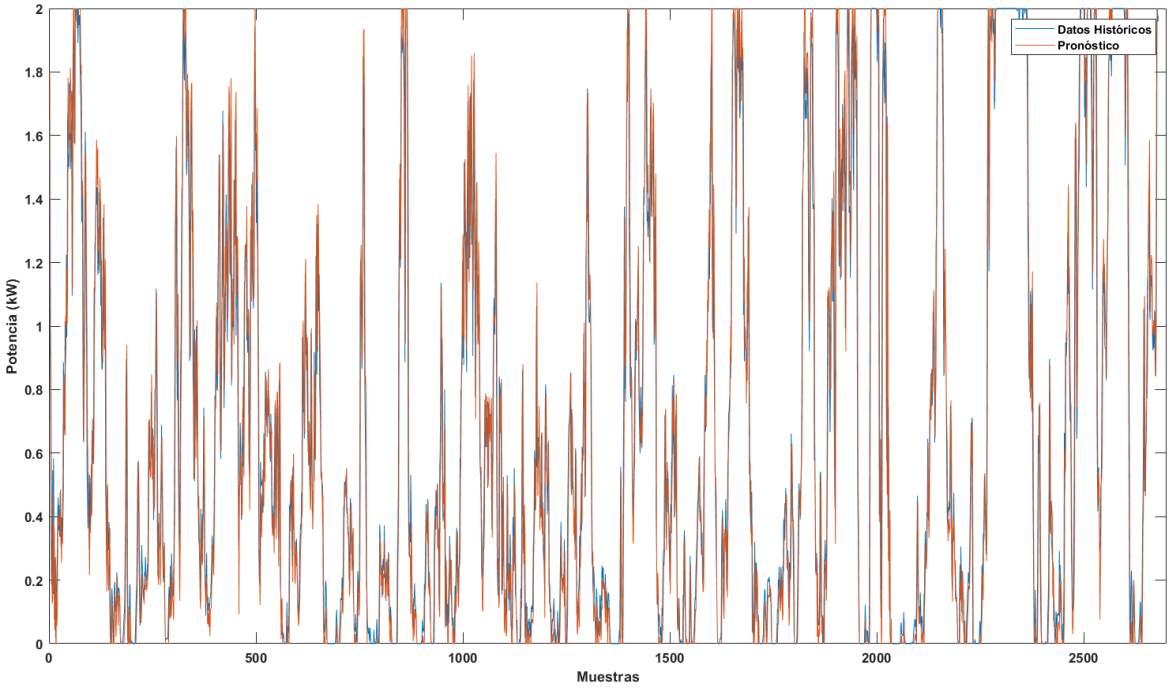

Figura 26 Resultado del entrenamiento de la evolución diferencial.

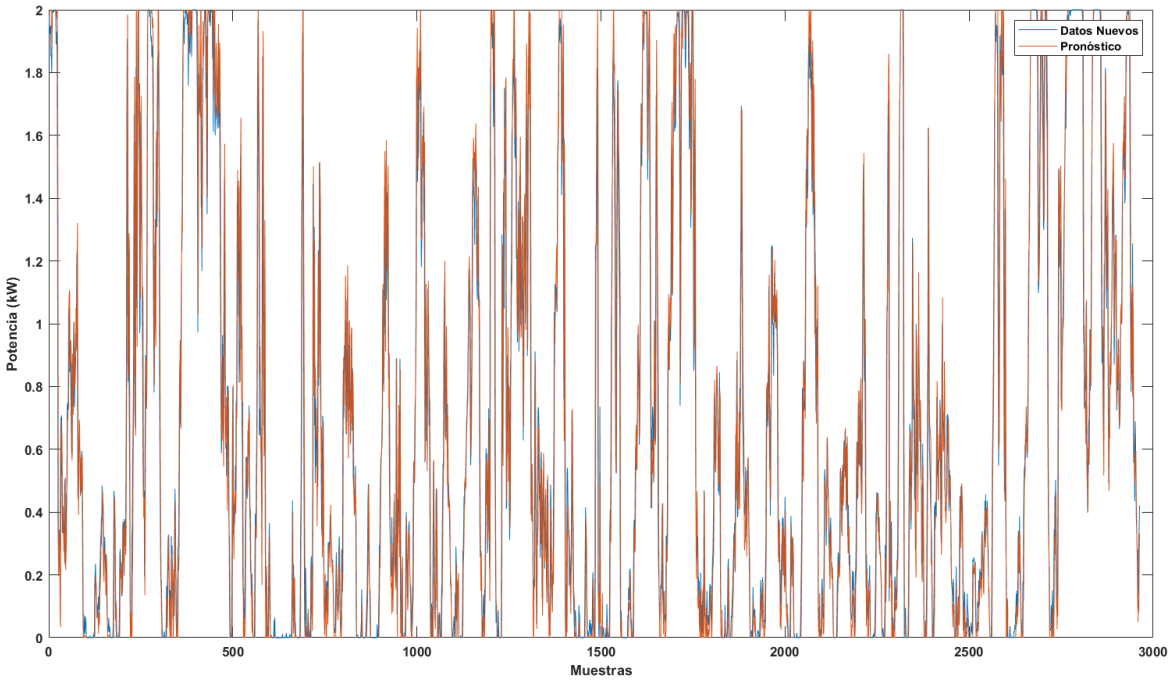

Figura 27 Resultado de la evaluación de la evolución diferencial.

En la Figura 26 y en la Figura 27 se muestra el resultado del entrenamiento y de la evaluación de la evolución diferencial respectivamente, se puede observar que la predicción sigue la tendencia de la curva tanto en el entrenamiento como en la evaluación, lo cual es de esperarse ya que se obtuvo una correlación de 0.964 y 0.96 respectivamente.

Los pesos obtenidos se muestran en la Tabla 6 y el modelo de predicción se muestra a continuación.

$$y_{k+1} = \sum_{i=1}^{n_a} w_i y_{k-i+1} + w_b$$

Donde:

y_{k+1} es la predicción,

w_i son los pesos,

y_{k-i+1} son los valores históricos,

w_b es el bias,

i es el retraso,

k es el momento actual, y

n_a es la cantidad de retrasos.

Tabla 6 Pesos obtenidos con evolucion diferencial.

i=1; 0.96742461880462	i=2; 0.17710034781217	i=3; -0.10992533908823
i=4; 0.00704694030925	i=5; -0.00810851670006	i=6; 0.08140579402309
i=7; 0.25112964348391	i=8; -0.29341327699100	wb -0.02388020059423

Predicción utilizando la red neuronal

La Figura 28 y la Figura 29 muestran el resultado del entrenamiento y la evaluación de la red neuronal respectivamente, se puede apreciar cómo ambas siguen la tendencia de la curva de potencia, sin embargo, se nota como el seguimiento de la curva es menor en la evaluación, lo cual es un resultado esperado ya que se obtuvo una correlación de 0.9733 en el entrenamiento y 0.962 en la evaluación.

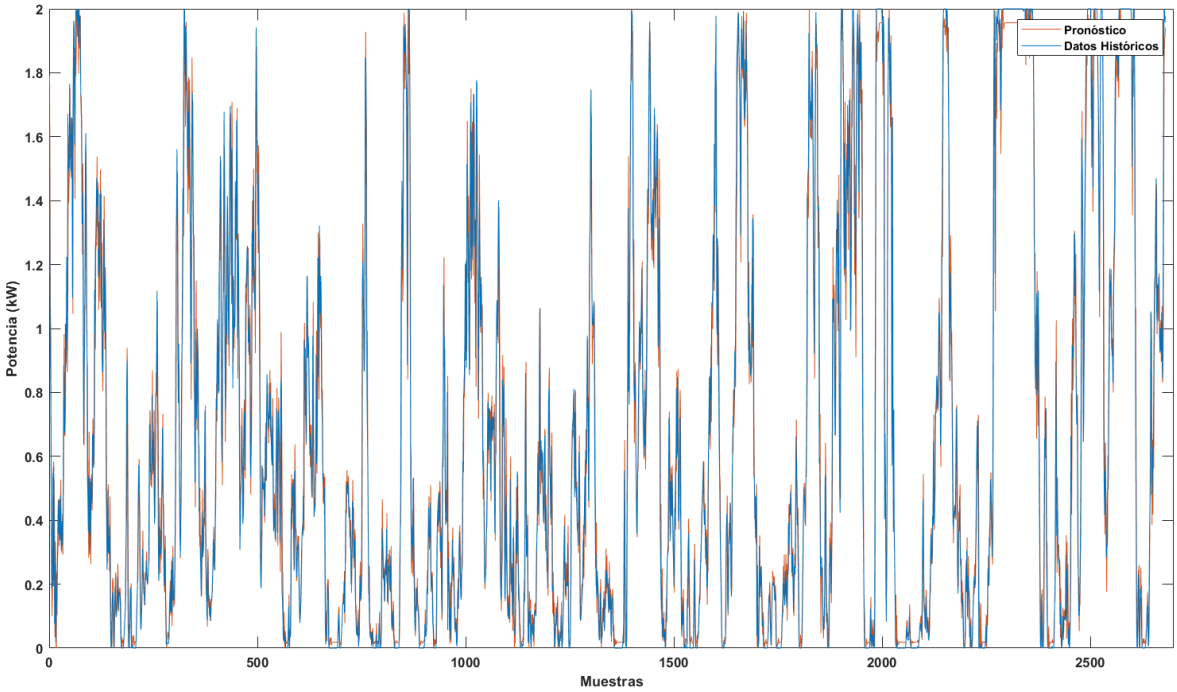

Figura 28 Resultado del entrenamiento de la red neuronal.

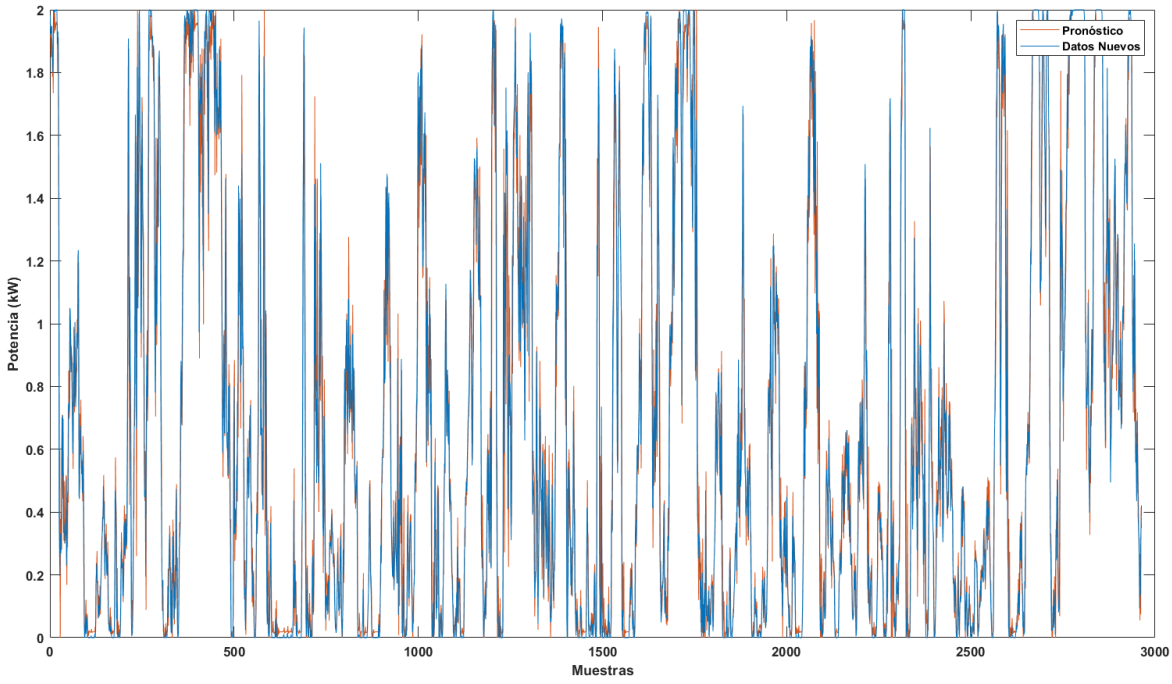

Figura 29 Evaluación de la red neuronal.

Comparación de los algoritmos de predicción

La primera diferencia a notar es en el tiempo de ejecución, un promedio de 0.6 segundos por época para la red neuronal y 0.67 segundos para la evolución diferencial. Para tener una mejor comparación entre algoritmos se presentan las gráficas de entrenamiento presentadas con anterioridad, pero con acercamiento a

24h, la Figura 30 corresponde a la ED y la Figura 31 corresponde a la RNA. De igual forma la Figura 32 y la Figura 33 muestran el acercamiento para la evaluación de los algoritmos, ED y RNA respectivamente, donde se aprecia el mejor desempeño de la RNA.

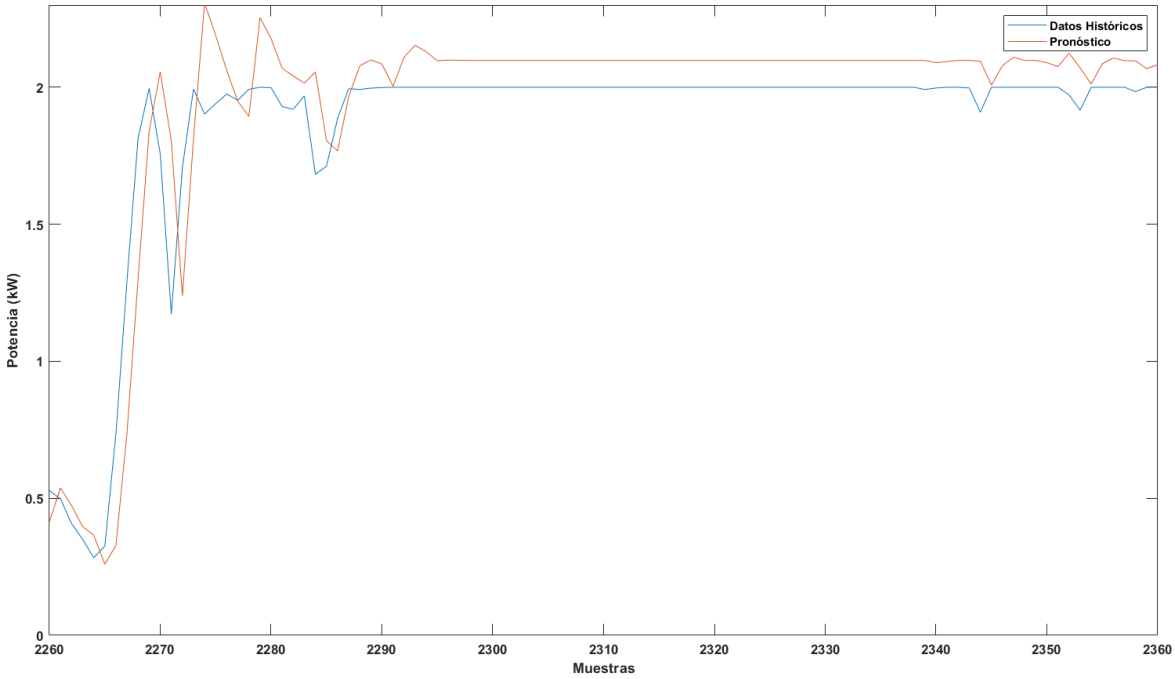

Figura 30 Resultados del entrenamiento de la ED, acercamiento a un día.

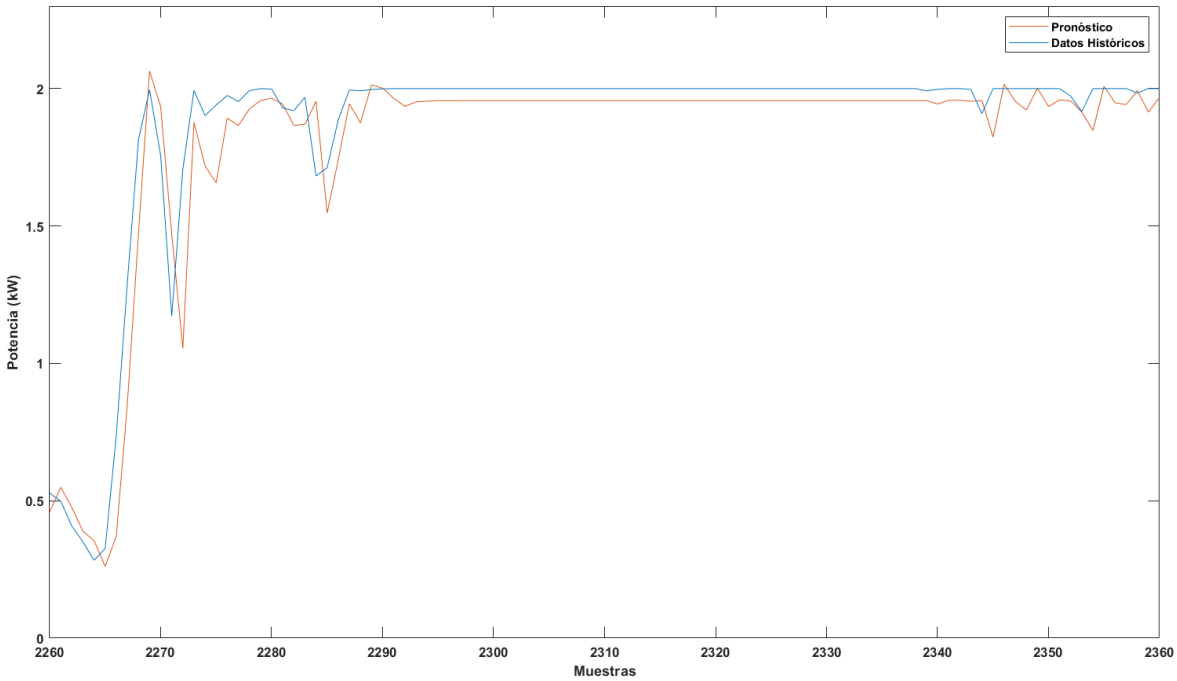

Figura 31 Resultados del entrenamiento de la RNA, acercamiento a 1 día.

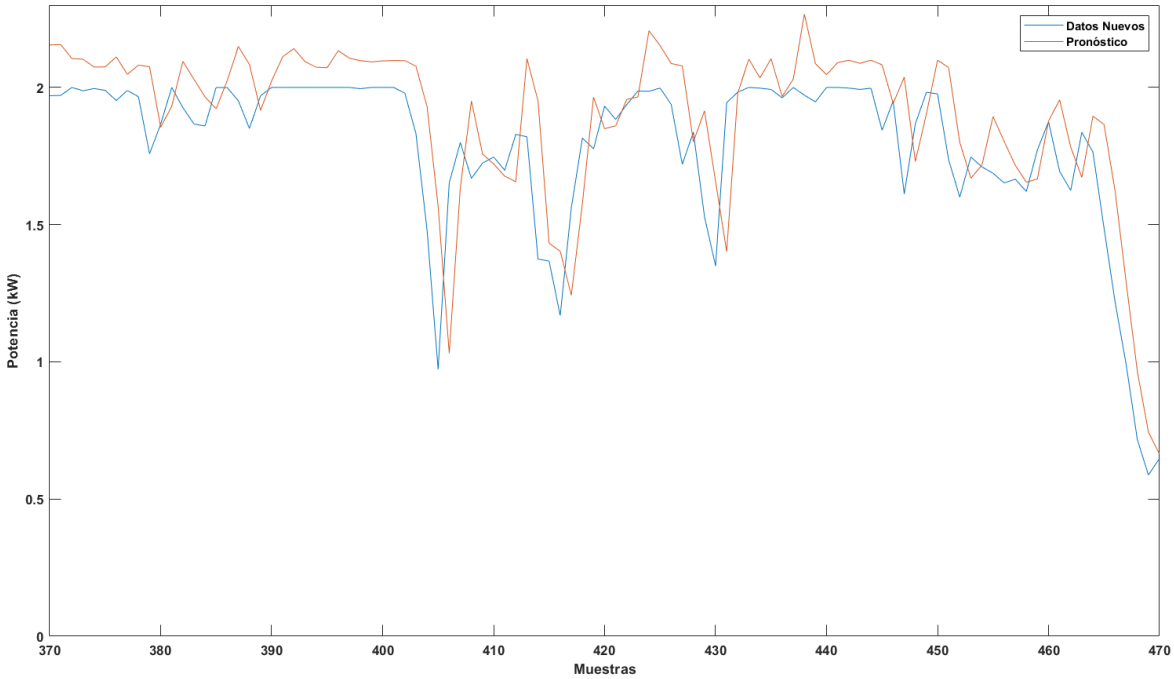

Figura 32 Evaluación de la ED, acercamiento a 1 día.

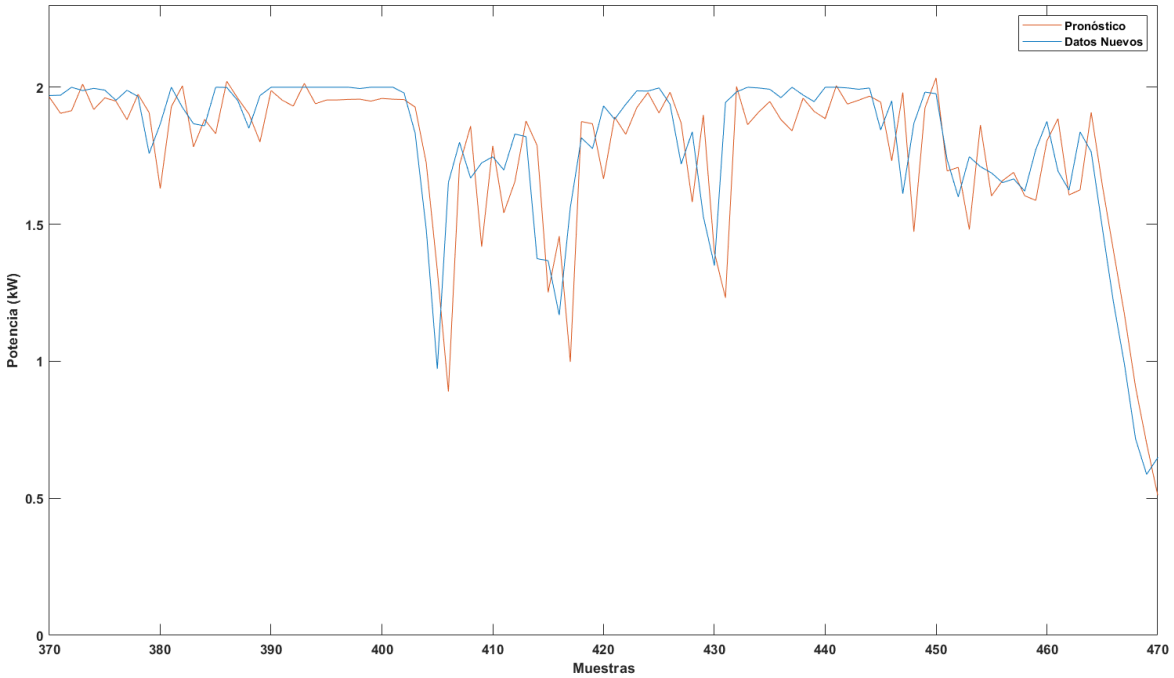

Figura 33 Evaluacion de la RNA, acercamiento a 1 día.

En la Tabla 7 se muestran los errores estadísticos de ambos algoritmos para el entrenamiento y en la

Tabla 8 se muestran los errores para la evaluación con datos no utilizados en el entrenamiento, con esta información se comprueba el mejor desempeño de la RNA ya que presenta un menor error, tanto en el entrenamiento como en la evaluación. El RMSE obtenido en el entrenamiento por la RNA es de 0.0746, mientras que el

de la ED es de 0.0946, en la evaluación se obtuvieron errores RMSE de 0.0904 para la RNA, y 0.1033 para la ED.

Tabla 7 Errores estadísticos de ambos algoritmos en el entrenamiento.

Error	**Red Neuronal**	**Evolución Diferencial**
RMSE	0.0746	0.0946
MSE	0.0056	0.0089
Correlación R	0.9733	0.9640

Tabla 8 Errores estadísticos de ambos algoritmos en la evaluación.

Error	**Red Neuronal**	**Evolución Diferencial**
RMSE	0.0904	0.1033
MSE	0.0082	0.0107
Correlación R	0.9620	0.9585

La RNA se ejecutó ligeramente más rápido que la ED, esto puede parecer anti intuitivo ya que en esencia la RNA es más compleja, sin embargo, para lograr tiempos de ejecución óptimos es necesario diseñar el software en base al hardware, por ejemplo si se cuenta con ALUs se debería usar números enteros, mientras, que si se tienen FPUs, se podría usar números de punto flotante, lo mismo sucede con las MACs, si se cuentan con ellas se puede acelerar el tiempo de ejecución y mejorar la precisión de operaciones que utilicen matrices, este es el caso de la mayoría de DSPs. Los errores estadísticos de ambos algoritmos fueron muy cercanos por lo que no me sorprendería que al cambiar los modelos se obtuviese un resultado invertido, donde la ED supere a la RNA.

Capítulo 5. Conclusiones

Se desarrollo un método de predicción de generación de potencia en sistemas eólicos basado en redes neuronales. Se utilizaron datos históricos de un parque eólico ubicado en Baja California. Se validó el método codificándolo en Matlab y, posteriormente, implementándolo en el DSP F28379D LaunchPad de Texas Instrument, para finalizar se comparó con el algoritmo de ED. Se logró mejorar sustancialmente el tiempo de ejecución del código previamente desarrollado en (Castro 2018) gracias a la implementación de operaciones matriciales y aprovechando las características del hardware. Se ha demostrado que la red neuronal usada en este estudio tiene la capacidad de producir una predicción con mayor certeza que el algoritmo de evolución diferencial. Basado en esta observación y dado que sus errores estadísticos son bajos, podemos concluir que una red neuronal representa un método útil para predecir la generación de potencia de las turbinas eólicas. En el trabajo presente solo se comparan una red neuronal con un número fijo de neuronas y el algoritmo base de evolución diferencial por lo que no se puede asumir que un algoritmo sea mejor de otro de forma general. Sin embargo, el resultado indica que la solución obtenida por la red neuronal es más robusta y con un menor error comparado con la evolución diferencial. Independientemente del resultado de este estudio, lo que determina el éxito de una red neuronal es la elección apropiada de los datos de entrenamiento y un razonable escalamiento de las variables de entrada. Se implementó exitosamente en hardware la red neuronal entrenada lo que permitiría la obtención de pronósticos rápidos sin la necesidad de una computadora, incluso podría implementarse en microcontroladores menos potentes y de menor consumo y lograr aun así tiempos de respuesta aceptables a un menor costo. Por otro lado, podría implementarse en un FPGA y lograr tiempos de ejecución más rápidos y un menor consumo de energía.

Recomendaciones para trabajos futuros

Existen múltiples variantes a los algoritmos utilizados, como la RNA LSTM o ELM, además de que hay muchos otros algoritmos de optimización que podrían dar mejores resultados a los obtenidos en este trabajo. Como la lógica difusa, árbol de decisiones o algoritmos genéticos.

Se mejoró el tiempo de ejecución la RNA por lo que podría ejecutarse en un microprocesador menos potente y más económico, lo que sería muy útil si se implementa como un producto comercial. Se recomienda probar con un Arduino ya que es una opción de bajo costo, tiene compatibilidad con simulink y podría no afectar tanto el tiempo de ejecución.

Referencias

Abbod, Maysam F. (2007). Application of Artificial Intelligence to the Management of Urological Cancer. “The Journal of Urology”, 178(4), 1150-1156.

Alanis A.Y., Ricalde, L.J., Sanchez E.N. (2009). High order neural networks for wind speed time series prediction. “2009 International Joint Conference on Neural Networks”, Atlanta, GA, 76-80.

Androutsos A.I., Papadopoulos M.P., Machias A.V. (1994). A logistic Model for on-line application in diesel-wind power systems. "Procd. of the 5th European Wind Energy Conference", EWEC’94, 2, 1068-1073.

Avila S. (2017). Contesting energy transitions: wind power and conflicts in the Isthmus of Tehuantepec. “Journal of Political Ecology”, 24(24).

Alexiadis M., Dokopoulos P., Sahsamanoglou H., Manousaridis I. (1998). Short term forecasting of wind speed and related electrical power. “Solar Energy”, 63(1), 61-68.

Armarol N, Balzani V. (2011). Towards an electricity-powered world. “Energy & Environmental Science”, 4(9), 3193.

Arriaga L., Espinoza J.M., Aguilar C., Martínez E., Gómez L. y Loa E. (2000). Regiones terrestres prioritarias de México. Comisión Nacional para el Conocimiento y uso de la Biodiversidad, México.

Aybay I., Etinkaya S., Halici U. (2000). Classification of neural network hardware, “Neural Network World”, IDG Co., 6(1), 11-29.

Balouktsis A., Tsanakas D., Vachtsevanos G. (1986). Stochastic simulation of hourly and daily average wind speed sequences. “Wind Engineering”, 10(1), 1-11.

Baratta D., Bo G.M., Caviglia D.D., Diotalevi F., Valle M. (1998). Microelectronic implementation of artificial neural network. En “Proceedings of the Fifth Electronics Devices and Systems International Conference”, Brno, Czech Republic.

Barbounis T.G., Theocharis J., Alexiadis M.C., Dokopoulos P.S. (2006). Long-term wind speed and power forecasting using local recurrent neural network models. “IEEE Transactions on Energy Conversion”, 21(1), 273-284.

Bechrakis D.A., Sparis P.D. (2004). Correlation of wind speed between neighboring measuring stations. “IEEE Trans Energy Convers”, 19(1), 400-6.

Belderbos A., Delarue E., (2015). Accounting for flexibility in power system planning with renewables. “Int J Electr Power Energy Syst” 71(1), 33-41.

Beyer H.G., Degner T., Haussmann J., Hoffman M., Rujan P. (1994). Short term prediction of wind speed and power output of a wind turbine with neural networks. En: “Proceedings of the Second European Congress on Intelligent Techniques and Soft Computing”, Aachen, Germany.

Billings S.A. (2013). Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains, 1ª edición, John Wiley & Sons, West Sussex, Reino Unido.

Bossanyi E.A. (1985). Short-term wind prediction using Kalman filters. “Wind Engineering”, 9(1), 1-8.

Broecker W. (1975). Climatic change: Are we on the brink of a pronounced global warming? “Science”, 189(1), 460- 463.

Broesch J.D., Stranneby D., Walker W. (2008). Digital Signal Processing: Instant access, 1ª edición. Butterworth-Heinemann-Newnes.

Calvert A.M., Bishop C.A., Elliot R.D., Krebs E.A., Kydd T.M., Machtans C.S., Robertson G.J. (2013). A synthesis of human-related avian mortality in Canada. “Avian Conservation and Ecology”, 8(2), 11.

Caviglia, D. (1999). NeuroNet Roadmap: Neural Networks Hardware. Universidad de Genova. Disponible en: http://clarkson.me.uk/academic/neuronet/NEuroNet_Roadmap_Neural_Networks_Hardware.htm Recuperado el 5 de Marzo de 2018.

Bishop C.M. (1995). Neural Networks for Pattern Recognition, 1ª edición, Oxford University Press, New York, Estados Unidos.

Da Y., Xiurun G. (2005). An improved PSO-based ANN with simulated annealing technique. “Neurocomputing”, 63(1), 527-533.

Damas M., Salmeron M., Díaz A., Ortega J., Prieto A., Olivares G. (2000). Genetic algorithms and neuro-dynamic programming: application to water supply networks. En: “Proceedings of 2000 Congress on Evolutionary Computation”, La Jolla, California, Estados Unidos.

Dasgupta D., Attoh-Okine N. (1997). Immunity-based systems: A survey. “Computational Cybernetics and Simulation”, 1(1), 369-374.

de Rigo D., Castelletti A., Rizzoli A. E., Soncini-Sessa R., Weber E. (2005). A selective improvement technique for fastening Neuro-Dynamic Programming in Water Resources Network Management. En: “Proceedings of the 16th IFAC World Congress”, Prague, Czech Republic, 16.

de Rigo D., Rizzoli A. E., Soncini-Sessa R., Weber E., Zenesi P. (2001). Neuro-dynamic programming for the efficient management of reservoir networks. En: “Proceedings of MODSIM 2001, International Congress on Modelling and Simulation” Canberra, Australia.

Deng G., Ferris M.C. (2008). Neuro-dynamic programming for fractionated radiotherapy planning. En: Alves C.J.S., Pardalos P.M., Vicente L.N. (eds) Optimization in Medicine. Springer Optimization and Its Applications, vol 12. Springer, New York, NY.

Dorigo M., Maniezzo V., Colorni A. (1996). Ant system: optimization by a colony of cooperating agents. “IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics”, 26(1), 29-41.

ELSAM. (1993). Wind power prediction tool in central dispatch centers. “Final report, EU Project JOU2-CT92-0083”.

Fabbri A., Gomezsanroman T., Rivierabbad J., Mendezquezada V.H. (2005). Assessment of the Cost Associated With Wind Generation Prediction Errors in a

Liberalized Electricity Market. "IEEE Transactions on Power Systems", 20(3), 1440-1446.

Ferreira C. (2006). Designing Neural Networks Using Gene Expression Programming. “Applied Soft Computing Technologies: The Challenge of Complexity”, 1(1), 517-536.

French Jordan. (2017). The time traveller’s CAPM. “Investment Analysts Journal”, 46 (2): 81-96.

Giebhardt R., Jochen T. (2010). Wind turbine condition monitoring systems and techniques. En “Wind Energy Systems: Optimizing design and construction for safe and reliable operation”, Capítulo 11. Elsevier, 329-349.

Gipe P. (1993). The Wind Industry’s Experience with Aesthetic Criticism. “Leonardo”, 26 (3), 243-248.

Global Wind Energy Council (GWEC). (2015). Global Wind Report - Annual market update 2015 Disponible en: http://www.gwec.net/wp-content/uploads/vip/GWEC-Global-Wind-2015-Report_April-2016_22_04.pdf Recuperado el 25 de febrero de 2017.

Bludszuweit H., Dominguez-Navarro J.A., and Llombart A. (2008) Statistical Analysis of Wind Power Forecast Error. “IEEE Transactions on Power Systems”, 23(3), 983-991.

Holttinen H., Meibom P., Orths A., Van Hulle F., Ensslin C., Hofmann L., McCann J., Pierik J., Olav Tande J., Estanqueiro A., Söder L., Strbac G., Parsons B., Smith C., Lemström B. (2016). Design and Operation of Power Systems with Large Amounts of Wind Power. “Agencia Internacional de la Energía".

Heemskerk, J.N. (1995). Neurocomputers for Brain-Style Processing. Design, Implementation and Application. Tesis de doctorado, Unit of Experimental and Theoretical Psychology Leiden University, The Netherlands.

Hoerner S. (1965). Sighard Hoerner's Fluid Dynamics. Hoerner's Fluid Dynamics.

Holler M. (1991). VLSI implementation of learning and memory systems: a review. “Advances in Neural Information Processing Systems”, 3, 993-1000.

Hoskins J.C., Himmelblau D.M. (1992). Process control via artificial neural networks and reinforcement learning. “Computers & Chemical Engineering”, 16 (4), 241-251.

Huang J., Lu X., McElroy M. (2014). Meteorologically defined limits to reduction in the variability of outputs from a coupled wind farm system in the Central US. “Renewable Energy”, 62, 331-40.

Hyndman R.J., Athanasopoulos G. (2013) Forecasting: principles and practice. Disponible en: https://otexts.com/fpp2/accuracy.html Recuperado el 15 de marzo de 2018.

Iten P.D. (1976). Laser Doppler Anemometer. “United States Patent and Trademark Office”. Disponible en: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/srchnum.html&r=1&f=G&l=50&d=PALL&s1=3966324.PN. Accedido el 18 de Septiembre de 2018.

Friedman J.H. (2002). Stochastic gradient boosting. “Computational Statistics and Data Analysis”, 38, 367-378.

Jesus C, (2018). Predicción de generación de potencia en instalaciones eólicas conectadas a la red mediante redes neuronales artificiales. Tesis Facultad de ingeniería de la UADY.

Kalogirou S.A. (2001). Artificial neural networks in renewable energy systems applications: a review. “Renew Sustain Energy”, 5, 373-401.

Kamal L., Jafri Y.Z. (1997). Time series models to simulate and forecast hourly averaged wind speed in Quetta, Pakistan. “Solar Energy”, 61(1), 23-32.

Kantz H., Holstein D., Ragwitz M., Vitanov N.K. (2004). Markov chain model for turbulent wind speed data. “Physica A: Statistical Mechanics and its Applications”, 342(2), 315-321.

Karaboga D. (2005). An idea based on honeybee swarm for numerical optimization. Technical report-tr06, Erciyes University, engineering faculty, computer-engineering department.

Kariniotakis G., Stavrakakis G., Nogaret E. (1996). A fuzzy logic and neural network based wind power model. En: “Proceedings of the 1996 European Wind Energy Conference”, Goteborg, Sweden, 596-599.

Kariniotakis G.N., Stavrakakis G.S., Nogaret E.F. (1996). Wind power forecasting using advanced neural networks models. “IEEE transactions on Energy conversion”, 11(4), 762-767.

Keulan E. van, Colak S., Withagen H., Hegt H. (1994). Neural networks hardware performance criteria, “Proceedings of the IEEE Conference on Neural Networks”, 3, 1885-1888.

Kusiak A., Zheng H.-Y., Song Z. (2009). Wind Farm Power Prediction: A Data-Mining Approach. "Wind Energy", 12(3), 275-293.

Lefton S.A., Besuner P. (2006). The Cost of Cycling Coal Fired Power Plants, “Coal Power Magazine”, Winter 2006, 16-20.

Liao Y., Neural networks in hardware: a survey, Disponible en: http://bit.csc.lsu.edu/~jianhua/shiv2.pdf Recuperado 15 de Marzo de 2018.

Lin L., Eriksson J.T., Vihriala H., Soderlund L. (1996) Predicting wind behavior with neural networks. En: “Proceedings of the 1996 European Wind Energy Conference”, Goteborg, Sweden, 655-658.

Lindsey C., Lindblad T., (1994). Review of hardware neural networks: a user’s perspective, En “Proceedings of the Third Workshop on Neural Networks: From Biology to High Energy Physics”, Isola d’Elba, Italy, 26-30.

Lorenz E.N. (1967). The nature and theory of the general circulation of the atmosphere, World Meteorological Organization.

Fugon L., Juban J., and Kariniotakis G. (2008). Data mining for Wind PowerForecasting. En “Proceedings of the European Wind Energy Conference EWEC’08”, Brusselas, Bélgica.

Forouzanfar M., Dajani H.R., Groza V.Z., Bolic M., Rajan S. (2010). Comparison of Feed-Forward Neural Network Training Algorithms for Oscillometric Blood Pressure Estimation.En “4th International Workshop on Soft Computing Applications”, Arad, Romania, 119-123.

Lange M. (2005). On the uncertainty of wind power predictions - Analysis of the forecast accuracy and statistical distribution of errors. “Journal of Solar Energy Engineering”,2(127), 177-194.

Negnevitsky M., Johnson P., Santoso S. (2007). Short-term wind power forecasting using hybrid intelligent systems. En “Proceedings of the IEEE Power Engineering Society General Meeting”, 1-4.

Negnevitsky M., Santoso S., Hatziargyriou N. (2006). Data mining and analysis techniques in wind power system applications: abridged. En: “Proceedings of the IEEE Power Engineering Society General Meeting”.

Duran M.J., Cros D., Riquelme J. (2007). Short-Term Wind Power Forecast Based on ARX Models. “Journal of Energy Engineering", 133(3), 172-180.

Milligan M., Miller A., Chapman F. (1995). Estimating the economic value of wind forecasting to utilities. En: “Windpower ‘95. American Wind Energy Association Conference”, 285-294.

Molina M. Rowland F.S. (1974). Stratospheric sink for chlorofluoromethanes: chlorine atom-catalysed destruction of ozone. Nature 249, 810-812.

Nedjah N., da Silva R.M., Mourelle L.M., da Silva M.V.C. (2009). Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs. “Neurocomputing”, 72(10-12), 2171-2179.

Jones N.F., Pejchar L., Kiesecker J.M. (2015). The Energy Footprint: How Oil, Natural Gas, and Wind Energy Affect Land for Biodiversity and the Flow of Ecosystem Services. “BioScience”, 65(3), 290-301.

Ojha V.K., Abraham A., Snášel V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. “Engineering Applications of Artificial Intelligence”, 60, 97-116.

Onwubolu G.C., Davendra D. (2009). Differential evolution: A handbook for global permutation-based combinatorial optimization. Springer Science & Business Media.

Parrore T.J., Miller R.G. (1985). Generalized exponential Markov and model output statistics: a comparative verification. “Monthly Weather Review”, 113, 1524-1541.

Pedersen E., Persson K. (2004). Perception and annoyance due to wind turbine noise—a dose–response relationship. “The Journal of the Acoustical Society of America”, 116(6), 3460-3470.

Platt R., Fitch-Roy O., Paul G. (2012). Beyond the bluster: Why wind power is an effective technology. Institute for Public Policy Research.

Bessa R., Miranda V., Gama J. (2009). Entropy and Correntropy against Minimum Square Error in Off-Line and On-Line 3-day ahead Wind Power Forecasting. “IEEE Transactions on Power Systems”, 24(4), 1657-1666.

Jursa R. (2007). Variable selection for wind power prediction using particle swarm optimization. En: “Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation”, London, England, U.K., 2059-2065.

Jursa R. (2007). Wind power prediction with different artificial intelligence models. En: “Proceedings of the European Wind Energy Conference EWEC’07”, Milan, Italy.

REN21. (2017). Renewables 2017. Global status report.

de la Cruz E., Ricalde L.J., Atoche E.J.R., Bassam A., Sanchez E. (2018). Forecast and Energy Management of a Microgrid with Renewable Energy Sources Using Artificial Intelligence. En: Intelligent Computing Systems. ISICS 2018. Communications in Computer and Information Science, vol 820. Springer, Cham, 81-96.

Rönkkönen J. (2009). Continuous Multimodal Global Optimization with Differential Evolution-Based Methods. Lappeenranta University of Technology.

Dominic S., Das R., Whitley D., Anderson C. (1991). Genetic reinforcement learning for neural networks. En: *“*IJCNN-91-Seattle International Joint Conference on Neural Networks”*,* Seattle, WA, USA, 2, 71-76.

Sathyajith M. (2006). Wind energy: Fundamentals, resource analysis and economics. Springer, Berlin and Heidelberg.

Schüffny R., Graupner A., Schreiter J. (1999). Hardware for neural networks. En: “Fourth International Workshop Neural Networks in Applications”, Magdeburg, Germany.

Secomandi N. (2000). Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. “Computers & Operations Research”, 27(11-12), 1201-1225.

Sepúlveda C.A., Muñoz J.A., Espinoza J.R., Figueroa M.E., Baier C.R. (2013). FPGA v/s DSP performance comparison for a VSC-based STATCOM control application. “IEEE Transactions on Industrial Informatics”, 9(3), 1351-1360.

Soler-Bientz R., Watson S., Infield D. (2009). Preliminary study of long-term wind characteristics of the Mexican Yucatán Peninsula. “Energy Conversion and Management”, 50, 1773-1780.

Storn R., Price K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. “Journal of global optimization”, 11(4), 341-359.

Szarka J. (2007). Wind Power in Europe: Politics, Business and Society. Springer.

Barbounis T.G., Theocharis J.B. (2006). Locally recurrent neural networks for long-term wind speed and power prediction. “Neurocomputing”, 69, 466-496.

Barbounis T.G., Theocharis J.B. (2006). Long-term wind speed and power forecasting using local recurrent neural network models. “IEEE Transactions on Energy Conversion”, 21(1), 273-284.

Tan C.L., Quah T.S., Teh H.H. Artificial neural network that models human decision making. “Computer”, 29(3), 64-70.

Teh H.H. (1995). Neural logic networks, a new class of neural networks. Singapore: World Scientific Publishing.

Ting Q., Chen Z., Li S., Xiang W., Li M. (2004). A Learning Algorithm of CMAC Based on RLS. “Neural Processing Letters”, 19, 49-61.

Ting Q., Chen Z., Li S., Xiang W. (2005). Continuous CMAC-QRLS and Its Systolic Array. “Neural Processing Letters”, 22, 1-16.

Farooq U., Marrakchi Z., Mehrez H. (2012). Tree Based Heterogeneous FPGA Architectures, Application Specific Exploration and Optimization. Springer.

Vapnik V.N. (1999). The Nature of Statistical Learning Theory, New York, Springer.

Walwyn D.R. Brent A.C. (2015). Renewable energy gathers steam in South Africa. “Renewable and Sustainable Energy Reviews”, 41, 390.

Watson S., Halliday J.A., Landberg L. (1992). Assessing the economic benefits of numerical weather prediction model wind forecasts to electric generating utilities. En “Proceedings of the Fourteenth British Wind Energy Association Conference”, 291-298.

Watson S., Halliday J.A., Landberg L. (1993). “Wind speed forecasting and its application to wind power integration”. En: “Proceedings of the Fifteenth British Wind Energy Association Conference”, 63-69.

Weigand A.S., Mangeas M., Srivastava A.N. (1995). Nonlinear gated experts for time series: discovering regimes and avoiding overfitting. "International Journal of Neural Systems", 6, 373-399.

Wu G., Dou Z. (1995). Nonlinear wind prediction using a fuzzy modular temporal neural network. En: “Windpower ‘95. American Wind Energy Association Conference”, 285-294.

Wu J., Chen E. (2009) A Novel Nonparametric Regression Ensemble for Rainfall Forecasting Using Particle Swarm Optimization Technique Coupled with Artificial Neural Network. En: Yu W., He H., Zhang N. (eds) Advances in Neural Networks. “Lecture Notes in Computer Science”, 5553. Springer, Berlin, Heidelberg

Wu Y., Ilong J. (2007). A literature review of wind forecasting technology in the world. “Power Tech 2007 IEEE Lausanne”, 504-507.

Apéndice A

Código Fuente en MATLAB de una red NARX. Este código se encuentra dentro del disco de la tesis bajo el nombre de “NN.mlx”. Antes de ejecutar el código es necesario tener en el mismo directorio el archivo de datos “Febrero_Agosto2.xlsx”, el cual se encuentra en el disco de la tesis.

```
Order=8;
Neurons=20;
Outputs=1;
MaxEpoch=100;
MaxError=.001;
P0=1000; %P inicial (dimension wixwi, wi=numero total de pesos)
Q0=20000; %Q inicial (dimension wixwi)
R0=25000;
Potencia= xlsread('Febrero_Agosto2.xlsx','PotProm','A2:A2689')';
Potencia=Potencia/1000;
Length=length(Potencia)-Order;
x=zeros(Order,Length);
for j=1:Order
   x(j,:)=Potencia(Order-j+1:Length+Order-j);
end
Target=Potencia(1,Order+1:Length+Order);
Target=Target/max(Target);
Input=x;
Epoch=0;
Error=100;
Inputs=size(Input,1);
WBiasElem=(Neurons*(Inputs+Outputs)+Neurons+Outputs);
Rowsbias=Inputs+1;
what=rand(WBiasElem,1)-0.5;
w1=reshape(what(1:Neurons*Rowsbias),Rowsbias,Neurons);
WeightsI=w1(1:Order,:);
Bias=w1(Rowsbias,:)';
WeightsO=what(Neurons*Rowsbias+1:WBiasElem);
P=P0*eye(WBiasElem); %P y Q matrices
Q=Q0*eye(WBiasElem);
R=R0*eye(Length);%2k
H1=ones(Length,Rowsbias*Neurons);
while Epoch<MaxEpoch && abs(Error)>MaxError
   %feedforward
   HiddenSum=WeightsI'*Input+Bias(1:Neurons,1);
   %funcion de Activacion
   S=logsig(HiddenSum);
```

```
  %train
  %Sumatoria de valores de capa previa por pesos siguientes
  OutputSum=WeightsO(1:Neurons)'*S+WeightsO(Neurons+1);
  %Activacion
  Output=OutputSum;
  %calculo de error lineal
  e=Target-Output;
  Error=sum(e.^2)/(2*Length)
  if Error>MaxError
     dsig2=logsig("dn",HiddenSum);%10*2k
     dsig2=WeightsO(1:Neurons).*dsig2;
     h=[Input' ones(Length,1)];
     for i=1:Length%2k
        for j=1:Neurons%10
           H1(i,(j-1)*Rowsbias+1:(j-1)*Rowsbias+Rowsbias)=h(i,:)*dsig2(j,i);
        end
     end
     H2=[S;ones(1,Length)];
     H=[H1';H2];
     % calculo de la ganancia de Kalman.
     K = P * H /(R + H' * P * H);
     % actualización de los pesos estimados.
     what = what + K * e';
     % actualización de la matriz de covarianza.
     P = P - K * H' * P + Q;
     w1=reshape(what(1:Neurons*Rowsbias),Rowsbias,Neurons);
     WeightsI=w1(1:Order,:);
     Bias=w1(Rowsbias,:)';
     WeightsO=what(Neurons*Rowsbias+1:WBiasElem);
  end
  Epoch=Epoch+1;
end
```

Apéndice B

Código Fuente en MATLAB de Evolución diferencial. Este código se encuentra dentro del disco de la tesis bajo el nombre de “ED.mlx”. Antes de ejecutar el código es necesario tener en el mismo directorio el archivo de datos “Febrero_Agosto2.xlsx”, el cual se encuentra en el disco de la tesis.

```
TS=xlsread('Febrero_Agosto2.xlsx','PotProm','A2:A2689')';
TS=TS/1000;
maxTS=max(TS);
NP=5;%Multiplo de Poblacion
F=.5;%Peso de la diferencia
CR=.5;%Tasa de combinacion
MaxI=100;%Iteraciones maximas
MaxE=.001;%error maximo
D=8;%Numero de retrasos
DD=D+1;%Numero de variables
DNP=D*NP;%Poblacion
G1=zeros(DNP,DD);
G=rand(DNP,DD);%inicializacion primera poblacion
it=0;%contador iteraciones
out=0;
foi=ones(1,DNP);
while it<MaxI && out==0
   for i=1:DNP
      a=randi(DNP);%i,a,b y c deben ser diferentes
      while (a==i)
         a=randi(DNP);
      end
      b=randi(DNP);
      while (b==i || b==a)
         b=randi(DNP);
      end
      c=randi(DNP);
      while (c==i || c==a || c==b)
         c=randi(DNP);
      end
      Forced=randi(DD);%por lo menos una variable se tiene que combinar
      v = zeros(1,DD);
      for j=1:DD
         if rand < CR || j==Forced
            v(j)=G(a,j) + F * (G(b,j) - G(c,j));
         else
            v(j)=G(i,j);
```

```
        end
    end
    fov=FO(TS,v);
    fog=FO(TS,G(i,:));
    if fov < fog
        G1(i,:)=v;
        foi(i)=fov;
    else
        G1(i,:)=G(i,:);
        foi(i)=fog;
    end
    if foi(i)<=MaxE
        out=1;
    end
  end
  it=it+1;
  G=G1;
end
%Encontrar el indice minimo
fomin=foi(1);
minIndex=1;
for i = 2:DNP
  if foi(i) < fomin
    minIndex = i;
    fomin = foi(i);
  end
end
Weights=G(minIndex,:);
function RMSE = FO(TS,Var)
  lvar=length(Var);
  lTS=length(TS);
  yhat=zeros(1,lTS-lvar);
  for i=lvar+1:lTS
    yhat(i-lvar)=Var(9)+Var(8)*TS(i-8)^8+Var(7)*TS(i-7)^7+Var(6)*TS(i-6)^6+Var(5)*TS(i-5)^5+Var(4)*TS(i-
4)^4+Var(3)*TS(i-3)^3+Var(2)*TS(i-2)^2+Var(1)*TS(i-1);
  end
  RMSE = sqrt(mean((TS(lvar+1:lTS) - yhat).^2));
end
```

Apéndice C

Para programar el F2837D con simulink es necesario tener instalados el ControlSUIT, el Code Composer Studio, el C2000Ware y el add-on de MATLAB “Embedded Coder Support Package for Texas Instruments C2000 Processors”.

Ya con los programas instalados se configura el add-on, para eso se abre MATLAB, en la pestaña de *HOME* se abre el menú expandible de *add-ons* y se hace click en *manage*, en la venta que se abre se le da click secundario a “*Embedded Coder Support Package for Texas Instruments C2000 Processors*” y lo configuramos presionando *setup*. Se nos abrirá la ventana de la Figura 34 donde se selecciona el modelo del procesador que se utilizará, en el caso de este trabajo F2837xD.

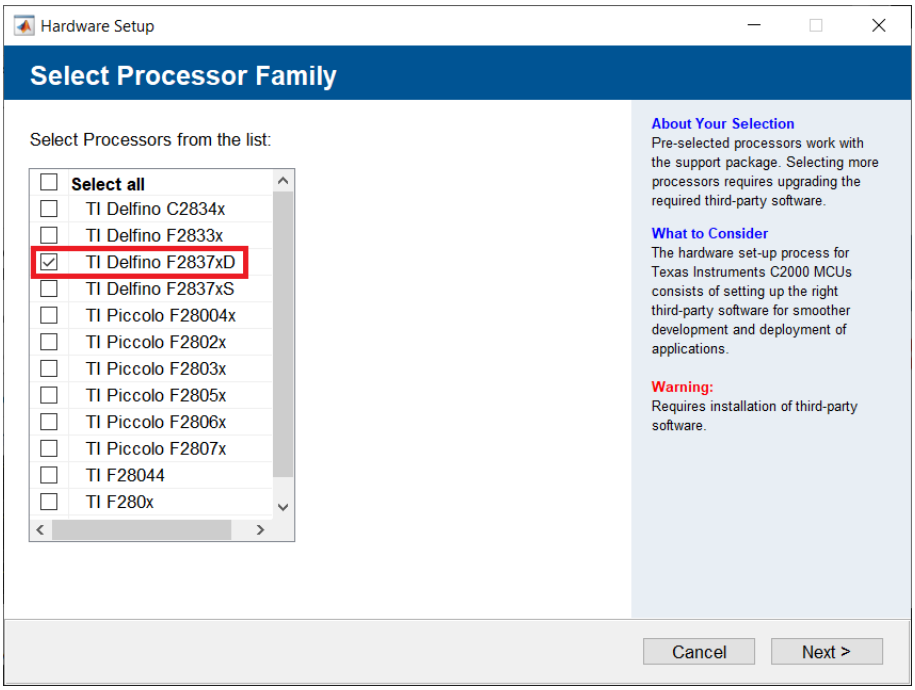

Figura 34 Se selecciona el modelo del hardware a utilizar.

Se da click en next y en la Figura 35, que es la siguiente ventana, se muestran los programas necesarios para programar el hardware y sí se detectó su instalación, sí se está seguro de que se tiene todo instalado se da click en next.

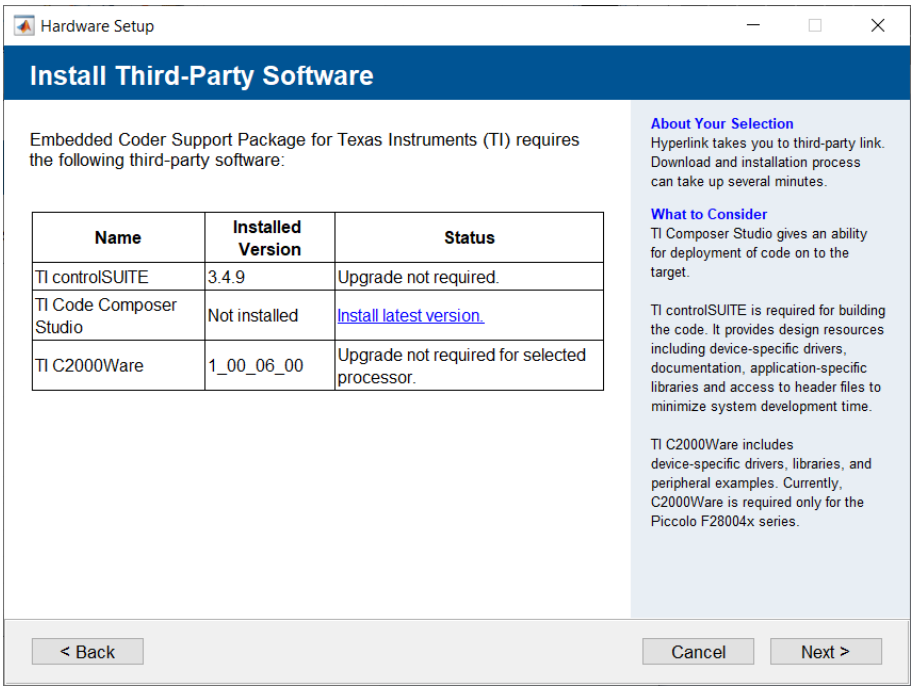

Figura 35 Software necesario para programar el Hardware.

En la Figura 36, se selecciona el directorio donde se instaló el ControlSuit, típicamente en el directorio que se muestra. Next.

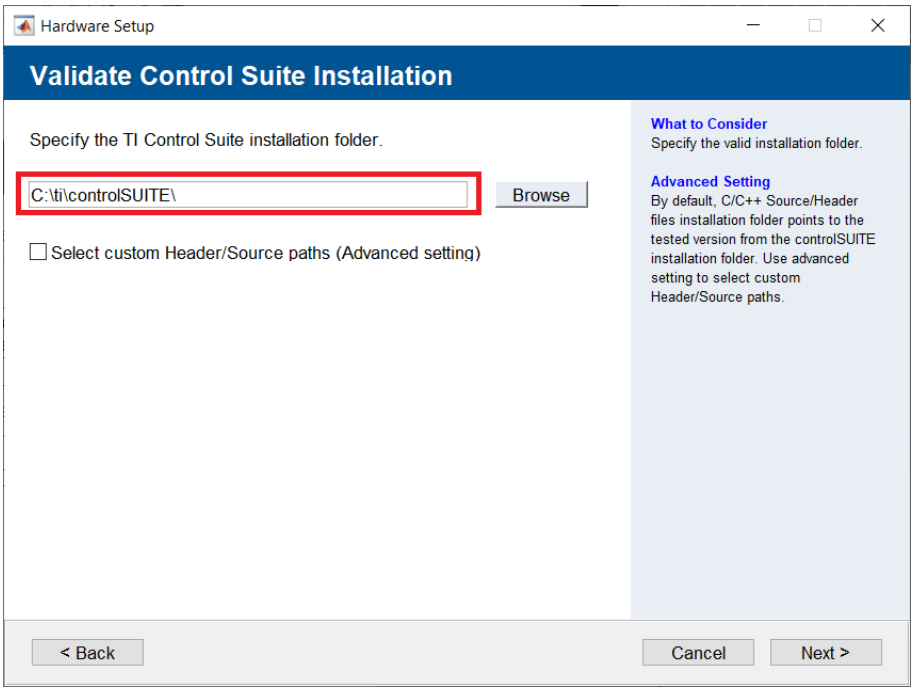

Figura 36 Selecciona la carpeta de instalación de Control Suite.

En la Figura 37 se selecciona el directorio del code composer studio y se da click en validar. A partir de la versión 9 se instala en un nuevo directorio, como se ve en la imagen. Next.

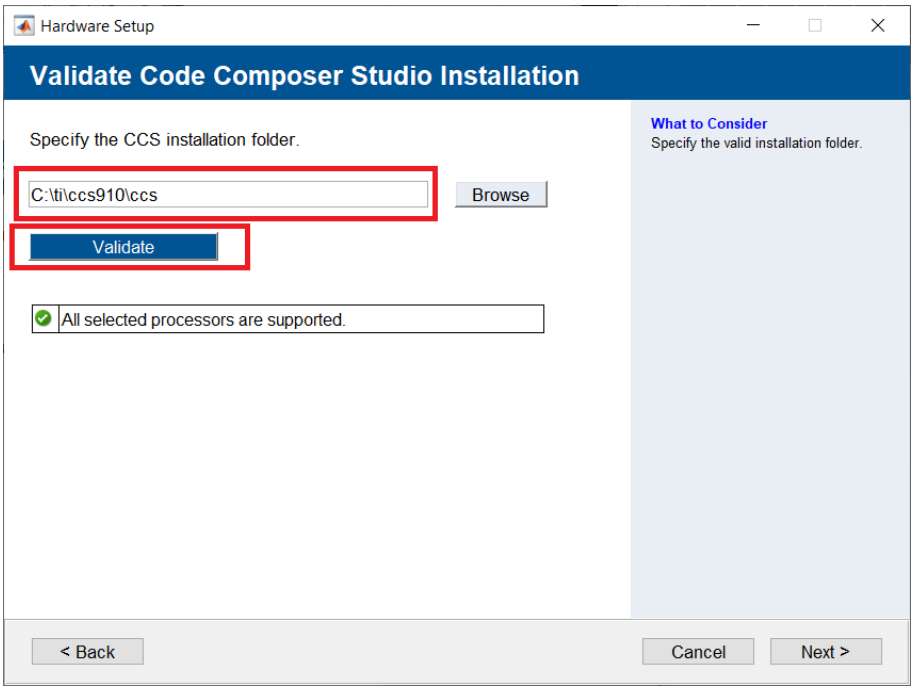

Figura 37 Selección de la carpeta de instalación de code composer studio.

En la Figura 38 se selecciona el directorio del compilador C200 que ya trae instalado MATLAB por default, Next.

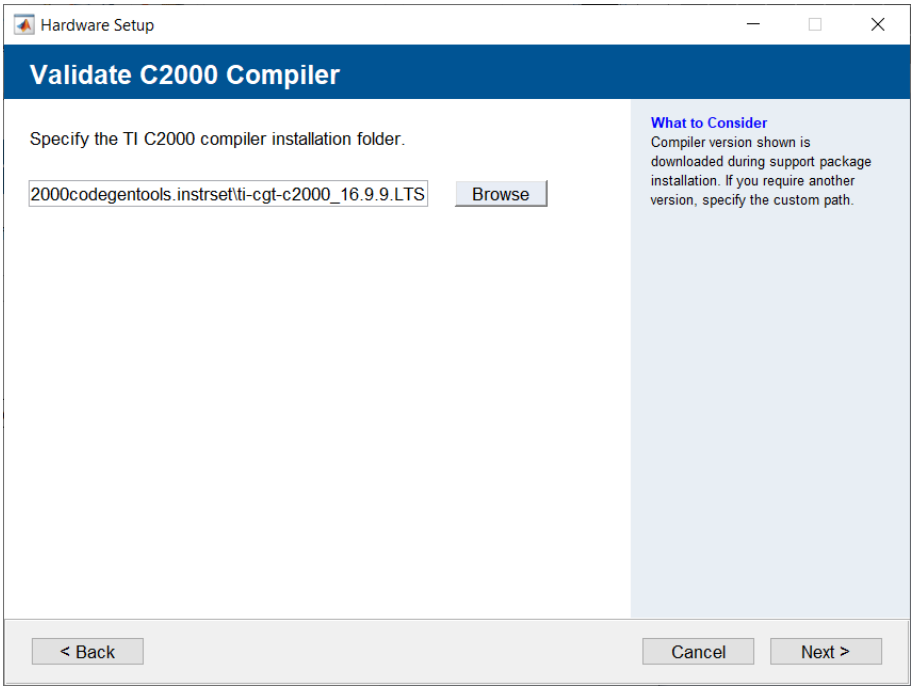

Figura 38 Selección de compilador. Valor predefinido.

Por último, nos muestra un resumen de la configuración que se seleccionó. Next y finish.

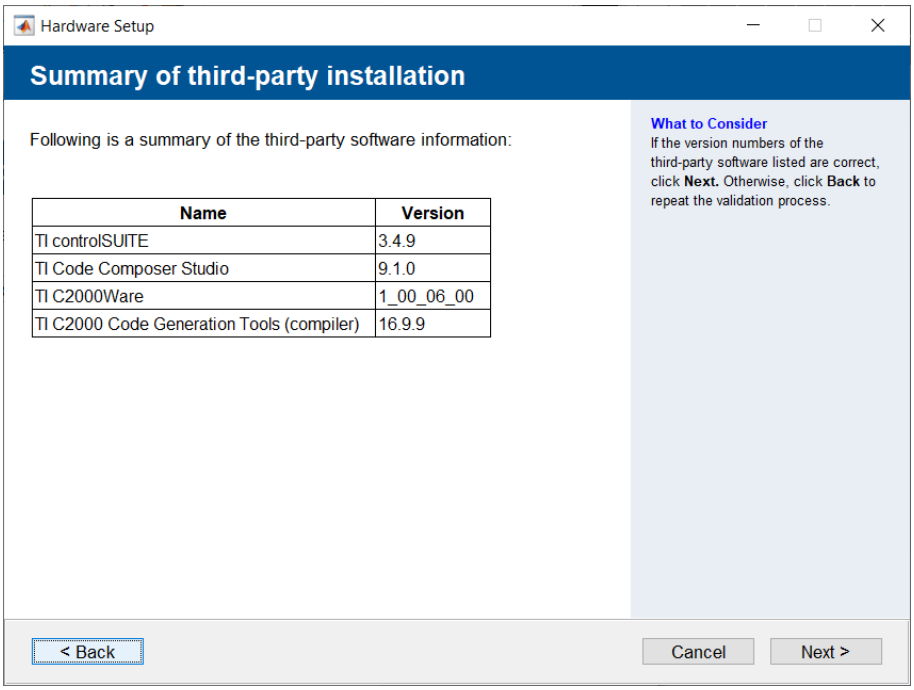

Figura 39 Resumen de la configuración hecha.

Con el software instalado y configurado se abre un nuevo archivo de simulink, y se agregan los bloques como se muestra en la Figura 40.

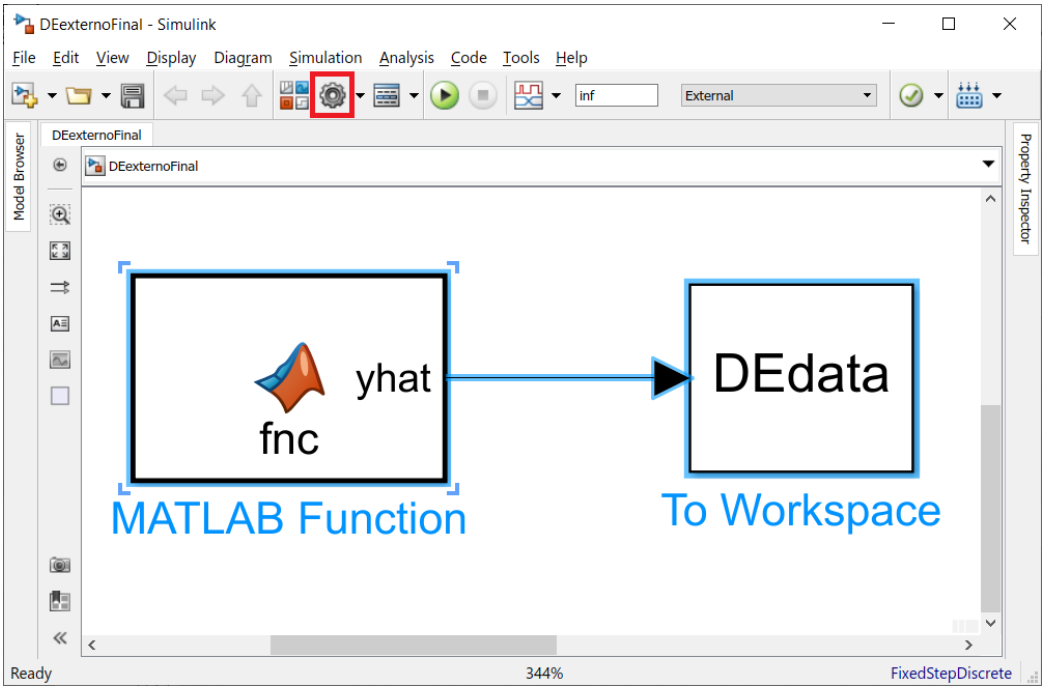

Figura 40 Modelo construido en simulink.

La configuración del bloque To Workspace se muestra en la Figura 41 y el código del bloque MATLAB Function se muestra al final de este apéndice.

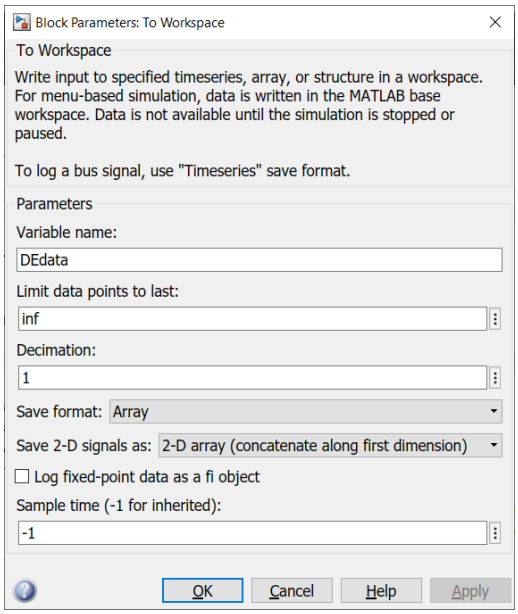

Figura 41 Ventana de configuración del bloque "To Workspace".

Ya armado el modelo hay que configurarlo, se hace click en el engrane de la Figura 40. En la ventana de configuración en la pestaña *Data Import/Export* se selecciona únicamente output y se selecciona el formato *Array* configura como en la Figura 42.

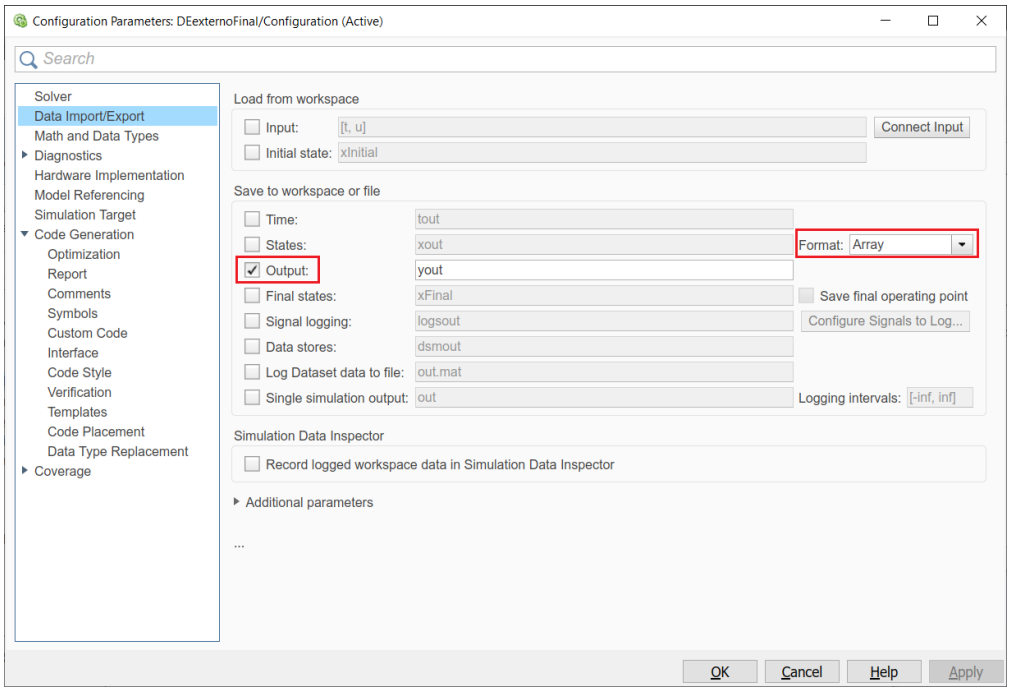

Figura 42 En Data Import/Export se selecciona únicamente output en un arreglo.

Posteriormente en la sección *Hardware Implementation* se selecciona el hardware con el que se trabajará y se le da click en *Target Hardware Resources* para que muestre la configuración avanzada. Esto se muestra en la Figura 43.

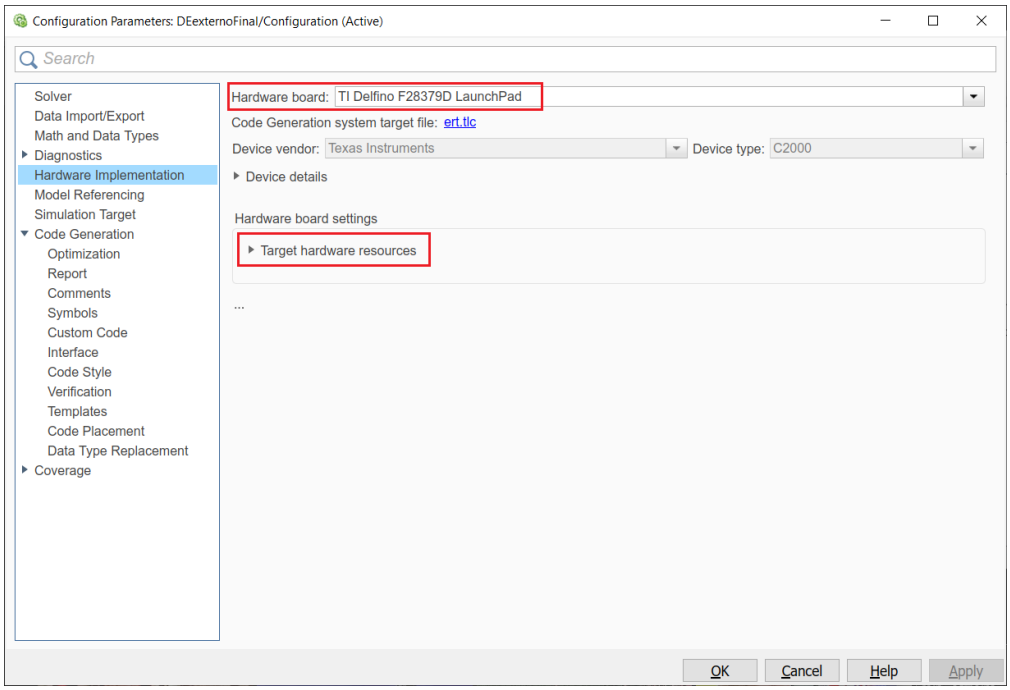

Figura 43 Se selecciona la tarjeta apropiada y expandimos el menú de recursos del hardware.

Lo siguiente es seleccionar *Use custom linker command file*, como se muestra en la Figura 44, debido a que se necesitan transmitir muchos datos y esto satura una parte de la memoria de la tarjeta, con el linker se administra manualmente la memoria y corrige este error.

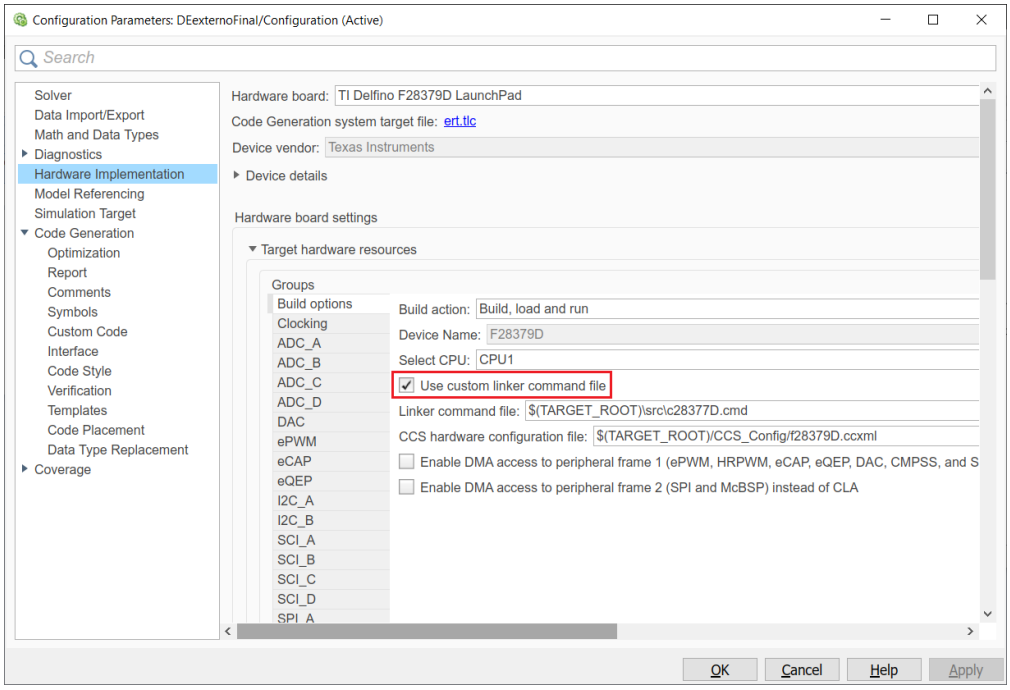

Figura 44 En el menú de recursos del hardware se selecciona usar linker personalizado

El paso siguiente es que dentro de la sección *External mode* configuramos el puerto de comunicaciones, como se muestra en la Figura 45 y la Figura 46.

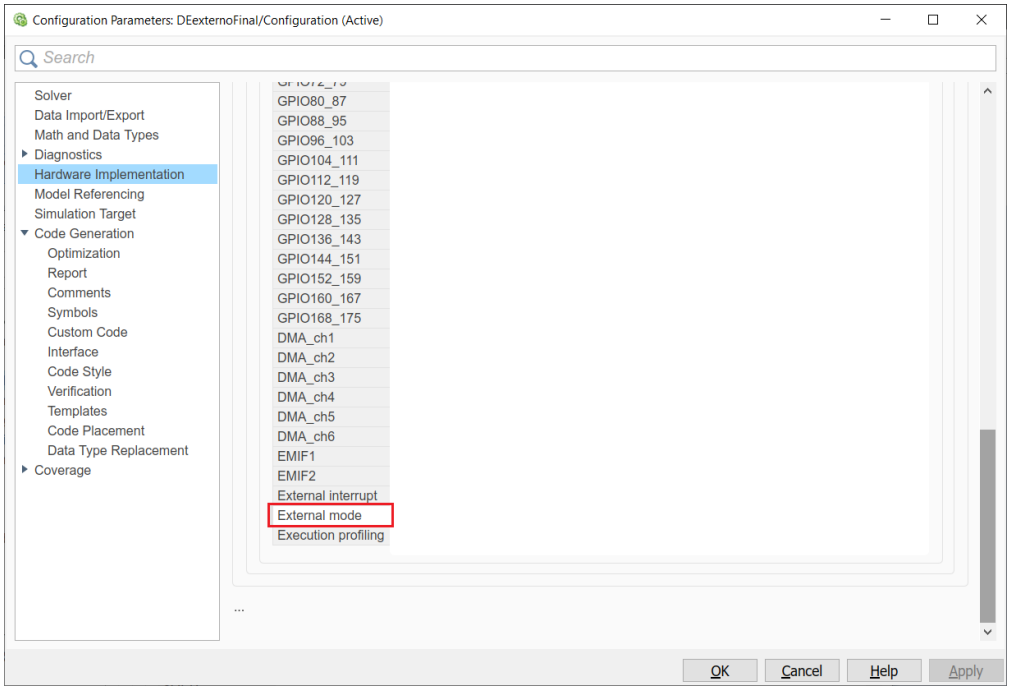

Figura 45 Se selecciona modo externo dentro del menú de recursos.

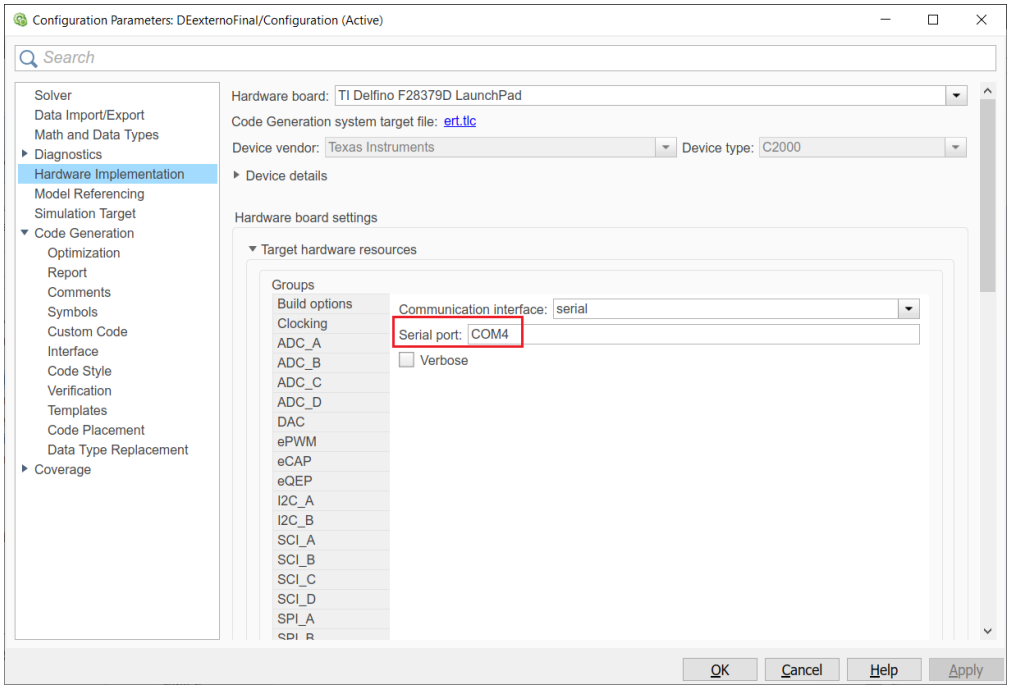

Figura 46 Se selecciona el puerto de comunicaciones que utiliza la tarjeta.

Para conocer el número del puerto se abre el administrador de dispositivos haciendo click secundario sobre el logo de Windows en la barra de tareas, como se observa en la Figura 47. En la Figura 48 se muestra la ubicación y nombre del

puerto de comunicaciones que es necesario introducir en la configuración de nuestro modelo.

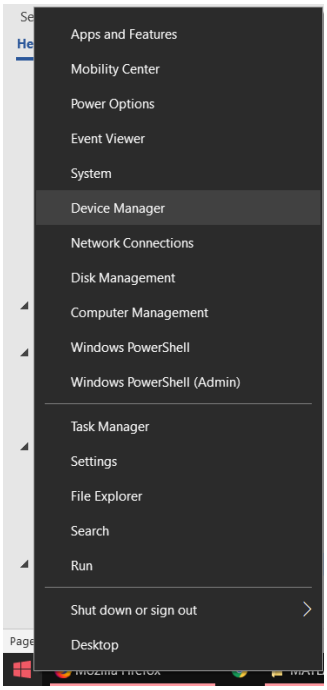

Figura 47 Seleccionar "administrador de dispositivos" al dar click secundario en inicio.

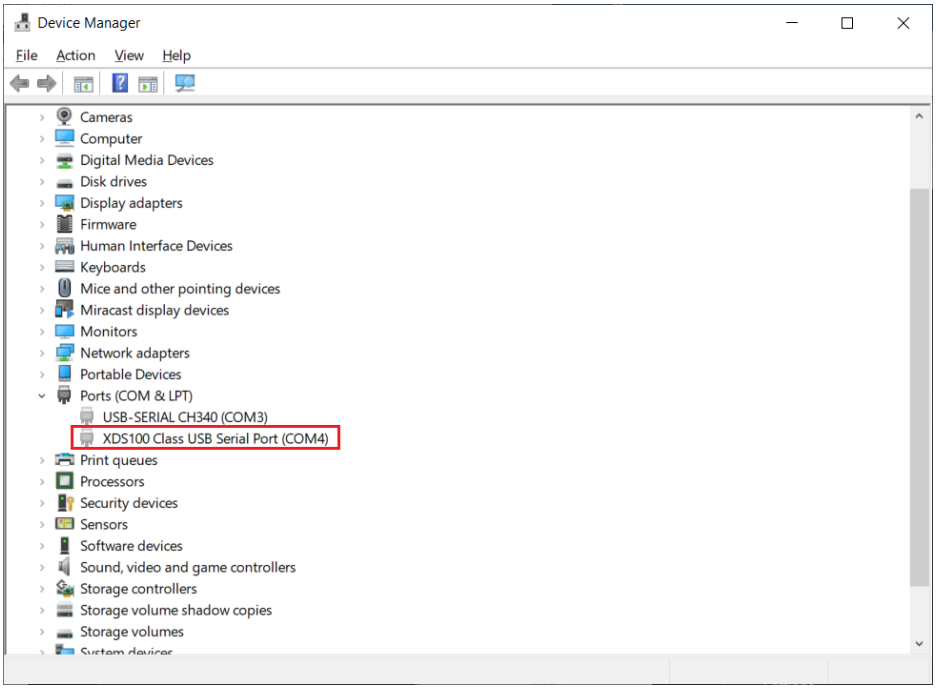

Figura 48 Expandir el árbol de los puertos y encontrar el que usa la tarjeta.

Por último, es necesario aumenta el tamaño de la cabecera, la cual se encarga de almacenar los datos a enviar a MATLAB. Para eso, dentro de la configuración del modelo, en la sección Code Generation en el valor de build configuration

seleccionamos specify y manualmente editamos el heap_size, 8000 es un valor apropiado para los casi 3 mil datos que se utilizaron. Esto se observa en la Figura 49.

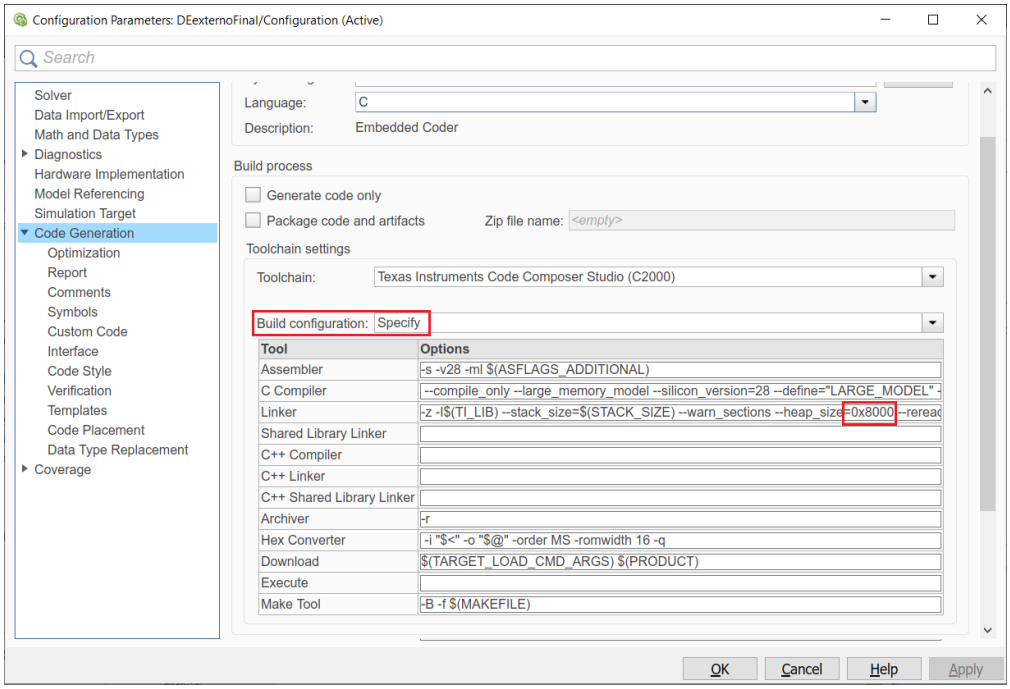

Figura 49 Modificar el tamaño del heap_size para poder enviar todos los datos.

Código Linker

En la línea 167 del archivo original incluido en MATLAB “c28377D.cmd” modificar la línea

```
.esysmem        : > RAMD0D1,    PAGE = 1
```

a

```
.esysmem        : > RAMGS | RAMD0D1,    PAGE = 1
```

En la línea 77 del archivo original incluido en MATLAB “c28377D.cmd” se agregó la siguiente línea, aunque realmente puede ir casi en cualquier línea siempre y cuando este antes de la línea donde se define esysmem.

```
RAMGS                   : origin = 0x015000, length = 0x007000
```

Para hacer estas modificaciones se necesitó revisar el memory map dentro del datasheet de la tarjeta.

Código del bloque MATLAB Function para la ED

```
function yhat  = fnc()
TS2=[0.993850000000000,0.991983333333335,…]; %serie de tiempo a pronosticar
Weights=[ 0.96742461880462,          0.17710034781217,          -0.10992533908823,
0.00704694030925,          -0.00810851670006,          0.08140579402309,
0.25112964348391,        -0.29341327699100,          -0.02388020059423];
D=8;%retrasos
yhat=zeros(1,length(TS2)-D);
for i=D+1:length(TS2)
   yhat(i-D)=Weights(9)+Weights(8)*TS2(i-8)^8+Weights(7)*TS2(i-7)^7+Weights(6)*TS2(i-
6)^6+Weights(5)*TS2(i-5)^5+Weights(4)*TS2(i-4)^4+Weights(3)*TS2(i-3)^3+Weights(2)*TS2(i-
2)^2+Weights(1)*TS2(i-1);
end
```

Código del bloque MATLAB Function para la RNA

```
function OutputSum2 = fnc()
Order=8;
Neurons=10;
Outputs=1;
WBiasElem=(Neurons*(Order+Outputs)+Neurons+Outputs);
Rowsbias=Order+1;
what=[];%Pesos
w1=reshape(what(1:Neurons*Rowsbias),Rowsbias,Neurons);
WeightsI=w1(1:Order,:);
Bias=w1(Rowsbias,:)';
WeightsO=what(Neurons*Rowsbias+1:WBiasElem);
% Length2=length(Potencia2)-Order;
Length2=2963;
x2=[]%8xdatasize matrix;
HiddenSum2=bsxfun(@plus,WeightsI'*x2,Bias(1:Neurons,1));
%funcion de Activacion
S2=1./(1+exp(-HiddenSum2));
%Sumatoria de valores de capa previa por pesos siguientes
OutputSum2=WeightsO'*[S2;ones(1,Length2)];
end
```