PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERÍA INFORMÁTICA

PRONOSTICO DE CAPTURAS DE ANCHOVETAS CON ALGORITMOS GENETICOS Y MAQUINAS DE VECTOR DE SOPORTE

NICOLÁS GUILLERMO CASELLI BENAVENTE

INFORME FINAL PARA OPTAR AL GRADO
ACADEMITO DE MAGISTER EN INGENIERIA
EN INFORMÁTICA

OCTUBRE 2011

DEDICATORIA

A mis padres y hermanos, por creer en todo momento en mí.

AGRADECIMIENTOS

A mis amigos y profesor guía, por todo el apoyo.

RESUMEN

El problema de pronóstico de captura de anchovetas en la zona norte de Chile tradicionalmente ha sido modelado utilizando máquinas de vector de soporte (SVM). Sin embargo, la técnica de SVM no considera cómo estimar los buenos parámetros al kernel. Por lo tanto, en esta memoria de título es propuesto un de predicción de captura de anchovetas basado en algoritmos genéticos para obtener los parámetros del kernel y traspasarlos a la máquina vectorial de soporte (AGSVM). La mejor topología encontrada durante la fase de evaluación ha sido un R^2 = 1.

ABSTRACT

The problem of forecasting anchovy catch in northern Chile have traditionally been modeled using support vector machines (SVM). However, the SVM technique does not consider how to estimate the good parameters to the kernel. Therefore, herein is proposed a title of anchovy catch prediction based on genetic algorithms for kernel parameters and transfer them to the support vector machine (AGSVM). The best topology found during the assessment phase has been a R2 = 1.

Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1. Introducción

En el presente informe, explicará parte del primer objetivo, que es la comprensión de los Algoritmos Genéticos (AGs, abreviado) y Maquinas de Vector Soporte (MVS). Mencionando una pequeña reseña de su historia y sus usos.

Se presentará, la temática sobre lo pertinente a las Máquinas Vectoriales, explicando su funcionamiento, para qué es mayormente el uso, y ver cuáles son los componentes que ayudan al funcionamiento de los mismos.

Expuesto concisamente, un algoritmo genético (o AG para abreviar) es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas. Dado un problema específico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, pero se suelen generar aleatoriamente.

Las máquinas de Vectores de Soporte (Support Vector Machine, por sus siglas en ingles SVM) son un nuevo sistema de aprendizaje el cual ha tenido un desarrollo muy significativo en los últimos años tanto en la generación de nuevos algoritmos como en las estrategias para su implementación. SVM es un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel, en el cual las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización el cual utiliza elementos de la teoría de generalización. SVM es un sistema para entrenar máquinas de aprendizaje lineal eficientemente tanto que para clasificación como para regresión se han encontrado muchas aplicaciones como clasificación de imágenes, reconocimiento de caracteres, detección de proteínas, clasificación de patrones, identificación de funciones, etc. A continuación describiremos el sistema SVM para la identificación de función es no lineales. Para lograr esto comenzaremos dando una breve introducción a las funciones kernel, la teoría de generalización y la teoría de optimización las cuales nos ayudaran a comprender mejor al sistema SVM, y por último se describirán los métodos de SVM para regresión.

La idea de este documento, es que permite, una vez entendido la temática de los algoritmos genéticos, lograr introducir el tema de las Máquinas Vectoriales para complementar el tema correspondiente a desarrollar y evaluar el rendimiento de un modelo de pronóstico basado en algoritmo genético y máquinas vectoriales para la administración de las capturas de anchovetas mensuales en la zona norte de Chile.

Teniendo en claro los temas usados para plantear el modelo, se explicará el modelo desarrollado como simulador para responder al objetivo principal, obtener un pronóstico de las capturas de anchovetas utilizando algoritmos genéticos y máquinas vectoriales. Para esto se explicará la

implementación de los métodos tanto para AG (algoritmo genético), como SVM (del inglés, *support vector machine*).

Luego de explicar el modelo, se realizaran se aplicará el modelo en pruebas con datos de 50 años, con esto se realizarán los entrenamientos al pronosticador, analizarán los resultados y, en base a eso, se realizarán las modificaciones a modo de ir ajustando el pronosticador. Este ajuste es de vital importancia, ya que será el ajuste el que nos lleve a tener resultados que sean aceptables para la captura de anchoveta, es decir, lograr pronósticos asertivos.

Capítulo 2. Introducción a los Algoritmos Genéticos (AGs)

De vez en cuando, los creacionistas acusan a la evolución de que carece de utilidad como teoría científica porque no produce beneficios prácticos y no tiene relevancia en la vida diaria. Sin embargo, tan sólo la evidencia de la biología demuestra que esta afirmación es falsa. Hay numerosos fenómenos naturales para los que la evolución nos ofrece un sólido fundamento teórico. Por nombrar uno, el desarrollo observado de la resistencia, a los insecticidas en las plagas de cultivos, a los antibióticos en las bacterias, a la quimioterapia en las células cancerosas, y a los fármacos antirretrovirales en virus como el VIH, es una consecuencia abierta de las leyes de la mutación y la selección, y comprender estos principios nos ha ayudado a desarrollar estrategias para enfrentarnos a estos nocivos organismos. El postulado evolutivo de la descendencia común ha ayudado al desarrollo de nuevos medicamentos y técnicas, al proporcionar a los investigadores una buena idea de con qué organismos deben experimentar para obtener resultados que probablemente serán relevantes para los seres humanos. Finalmente, el hombre ha utilizado con grandes resultados el principio de cría selectiva para crear organismos personalizados, distintos a cualquiera que se pueda encontrar en la naturaleza, para beneficio propio. El ejemplo canónico, por supuesto, es la diversidad de variedades de perros domésticos (razas tan diversas como los bulldogs, chihuahuas y dachshunds han sido producidas a partir de lobos en sólo unos pocos miles de años), pero ejemplos menos conocidos incluyen al maíz cultivado (muy diferente de sus parientes salvajes, que carecen de las familiares “orejas” del maíz cultivado), a los peces de colores (como los perros, hemos criado variedades cuyo aspecto es drásticamente distinto al del tipo salvaje), y a las vacas lecheras (con ubres inmensas, mucho mayores que las necesarias para alimentar a una cría).

Los críticos pueden argumentar que los creacionistas pueden explicar estas cosas sin recurrir a la evolución. Por ejemplo, a menudo los creacionistas explican el desarrollo de la resistencia a los agentes antibióticos en las bacterias, o los cambios forjados en los animales domésticos por selección artificial, asumiendo que Dios decidió crear a los organismos en grupos fijos, llamados “tipos” o baramins. Aunque la microevolución natural o la selección artificial dirigida por humanos pueden producir diferentes variedades dentro de los “tipo-perro”, “tipo-vaca” o “tipo-bacteria” creados originalmente, ninguna cantidad de tiempo o cambio genético puede transformar un “tipo” en otro. Sin embargo, nunca se explica cómo determinan los creacionistas lo que es un “tipo”, o qué mecanismo impide a los seres vivos evolucionar más allá de sus límites.

Pero en las últimas décadas, el continuo avance de la tecnología moderna ha producido algo nuevo. Ahora la evolución está produciendo beneficios prácticos en un campo muy distinto y, esta vez, los creacionistas no pueden afirmar que su explicación se adapte a los hechos igual de bien. Este campo es la informática, y los beneficios provienen de una estrategia de programación llamada algoritmos genéticos. Este ensayo explicará qué son los algoritmos genéticos y mostrará de qué manera son relevantes en el debate evolución/creacionismo.

2.1 ¿Qué es un algoritmo genético?

Expuesto concisamente, un algoritmo genético (o AG para abreviar) es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas. Dado un problema específico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, pero se suelen generar aleatoriamente.

Luego el AG evalúa cada candidata de acuerdo con la función de aptitud. En un acervo de candidatas generadas aleatoriamente, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras -pueden mostrar actividad, aunque sólo sea actividad débil e imperfecta, hacia la solución del problema.

Estas candidatas prometedoras se conservan y se les permite reproducirse. Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios aleatorios durante el proceso de copia. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo acervo de soluciones candidatas, y son sometidas a una ronda de evaluación de aptitud. Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas de nuevo; pero, de nuevo, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. De nuevo, se selecionan y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios, y el proceso se repite. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y, por tanto, repitiendo este proceso cientos o miles de rondas, pueden descubrirse soluciones muy buenas del problema.

Aunque a algunos les puede parecer asombroso y antiintuitivo, los algoritmos genéticos han demostrado ser una estrategia enormemente poderosa y exitosa para resolver problemas, demostrando de manera espectacular el poder de los principios evolutivos. Se han utilizado algoritmos genéticos en una amplia variedad de campos para desarrollar soluciones a problemas tan difíciles o más difíciles que los abordados por los diseñadores humanos. Además, las soluciones que consiguen son a menudo más eficientes, más elegantes o más complejas que nada que un ingeniero humano produciría. ¡En algunos casos, los algoritmos genéticos han producido soluciones que dejan perplejos a los programadores que escribieron los algoritmos en primera instancia!

2.2 Métodos de representación

Antes de que un algoritmo genético pueda ponerse a trabajar en un problema, se necesita un método para codificar las soluciones potenciales del problema de forma que una computadora pueda procesarlas. Un enfoque común es codificar las soluciones como cadenas binarias: secuencias de 1s y 0s, donde el dígito de cada posición representa el valor de algún aspecto de la solución. Otro método similar consiste en codificar las soluciones como cadenas de enteros o números decimales, donde cada posición, de nuevo, representa algún aspecto particular de la solución. Este método permite una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios, y a menudo "está intuitivamente más cerca del espacio de problemas" (Fleming y Purshouse 2002[3], p 1.228).

Esta técnica se utilizó, por ejemplo, en el trabajo de Steffen Schulze-Kremer, que escribió un algoritmo genético para predecir la estructura tridimensional de una proteína, basándose en la secuencia de aminoácidos que la componen (Mitchell 1996[47], p. 62). El AG de Schulze-Kremer utilizaba números reales para representar los famosos "ángulos de torsión" entre los enlaces peptídicos que conectan a los aminoácidos. (Una proteína está formada por una secuencia de bloques básicos llamados aminoácidos, que se conectan como los eslabones de una cadena. Una vez que todos los aminoácidos están enlazados, la proteína se dobla formando una compleja estructura tridimensional, basada en cuáles aminoácidos se atraen entre ellos y cuáles se repelen. La forma de una proteína determina su función). Los algoritmos genéticos para entrenar a las redes neuronales también utilizan a menudo este método de codificación.

Un tercer método consiste en representar a los individuos de un AG como cadenas de letras, donde cada letra, de nuevo, representa un aspecto específico de la solución. Un ejemplo de esta técnica es el método basado en "codificación gramática" de Hiroaki Kitano, en el que a un AG se le encargó la tarea de evolucionar un sencillo conjunto de reglas llamadas gramática libre de contexto, que a su vez se utilizaban para generar redes neuronales para una variedad de problemas (Mitchell 1996[47], p. 74).

La virtud de estos tres métodos es que facilitan la definición de operadores que causen los cambios aleatorios en las candidatas seleccionadas: cambiar un 0 por un 1 o viceversa, sumar o restar al valor de un número una cantidad elegida al azar, o cambiar una letra por otra. (Ver la sección sobre los métodos de cambio para más detalles acerca de los operadores genéticos). Otra estrategia, desarrollada principalmente por John Koza, de la Universidad de Stanford, y denominada programación genética, representa a los programas como estructuras de datos ramificadas llamadas árboles (Koza et al. 2003[42], p. 35). En este método, los cambios aleatorios pueden generarse cambiado el operador o alterando el valor de un cierto nodo del árbol, o sustituyendo un subárbol por otro.

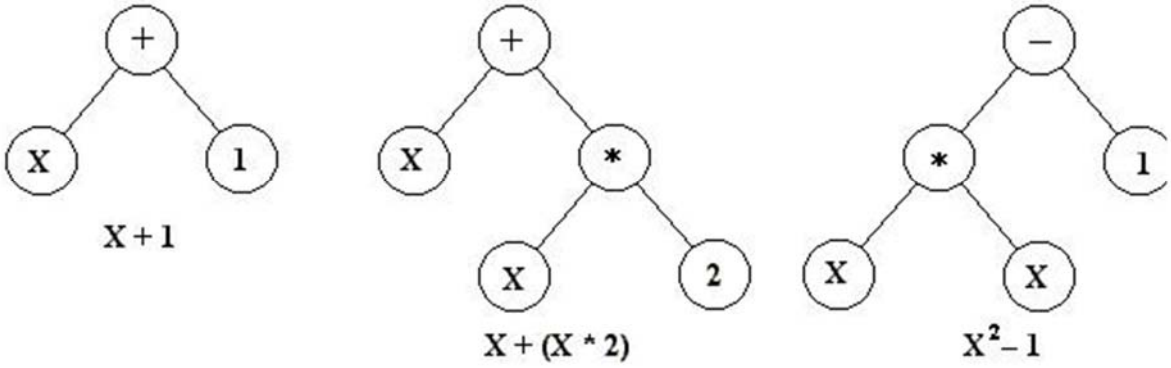

Figura 2-1 Tres sencillos árboles utilizado la programación genética

Aquí se muestran tres sencillos árboles de programa del tipo utilizado normalmente en la programación genética. Debajo se proporciona la expresión matemática que representa cada uno.

2.3 Métodos de selección

Un algoritmo genético puede utilizar muchas técnicas diferentes para seleccionar a los individuos que deben copiarse hacia la siguiente generación, pero abajo se listan algunos de los más comunes. Algunos de estos métodos son mutuamente exclusivos, pero otros pueden utilizarse en combinación, algo que se hace a menudo.

- Selección elitista: se garantiza la selección de los miembros más aptos de cada generación. (La mayoría de los AGs no utilizan elitismo puro, sino que usan una forma modificada por la que el individuo mejor, o algunos de los mejores, son copiados hacia la siguiente generación en caso de que no surja nada mejor).

- Selección proporcional a la aptitud: los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza.

- Selección por rueda de ruleta: una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores. (Conceptualmente, esto puede representarse como un juego de ruleta -cada individuo obtiene una sección de la ruleta, pero los más aptos obtienen secciones mayores que las de los menos aptos. Luego la ruleta se hace girar, y en cada vez se elige al individuo que “posea” la sección en la que se pare la ruleta).

- Selección escalada: al incrementarse la aptitud media de la población, la fuerza de la presión selectiva también aumenta y la función de aptitud se hace más discriminadora. Este método puede ser útil para seleccionar más tarde, cuando todos los individuos tengan una aptitud relativamente alta y sólo les distingan pequeñas diferencias en la aptitud.

- Selección por torneo: se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Sólo se elige a un individuo de cada subgrupo para la reproducción.

- Selección por rango: a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este ranking, en lugar de las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.

- Selección generacional: la descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación. No se conservan individuos entre las generaciones.

- Selección por estado estacionario: la descendencia de los individuos seleccionados en cada generación vuelven al acervo genético preexistente, reemplazando a algunos de los miembros menos aptos de la siguiente generación. Se conservan algunos individuos entre generaciones.

- Selección jerárquica: los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente. La ventaja de este método es que reduce el tiempo total de cálculo al utilizar una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que se muestran poco o nada prometedores, y sometiendo a una evaluación de aptitud más rigurosa y computacionalmente más costosa sólo a los que sobreviven a esta prueba inicial.

2.4 Métodos de cambio

Una vez que la selección ha elegido a los individuos aptos, éstos deben ser alterados aleatoriamente con la esperanza de mejorar su aptitud para la siguiente generación. Existen dos estrategias básicas para llevar esto a cabo. La primera y más sencilla se llama mutación. Al igual que una mutación en los seres vivos cambia un gen por otro, una mutación en un algoritmo genético también causa pequeñas alteraciones en puntos concretos del código de un individuo.

El segundo método se llama cruzamiento, e implica elegir a dos individuos para que intercambien segmentos de su código, produciendo una "descendencia" artificial cuyos individuos son combinaciones de sus padres. Este proceso pretende simular el proceso análogo de la recombinación que se da en los cromosomas durante la reproducción sexual. Las formas comunes de cruzamiento incluyen al cruzamiento de un punto, en el que se establece un punto de intercambio en un lugar aleatorio del genoma de los dos individuos, y uno de los individuos contribuye todo su código anterior a ese punto y el otro individuo contribuye todo su código a partir de ese punto para producir una descendencia, y al cruzamiento uniforme, en el que el valor de una posición dada en el genoma de la descendencia corresponde al valor en esa posición del genoma de uno de los padres o al valor en esa posición del genoma del otro padre, elegido con un 50% de probabilidad.

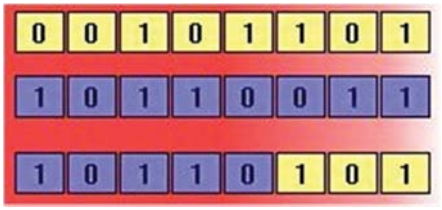

Figura 2-1 Ejemplo de cruce entre padres

Figura 2-2 Ejemplo de mutación.

Figura: Cruzamiento y mutación. El diagrama de arriba ilustra el efecto de estos dos operadores genéticos en los individuos de una población de cadenas de 8 bits. El diagrama superior muestra a dos individuos llevando a cabo un cruzamiento de un punto; el punto de intercambio se establece entre las posiciones quinta y sexta del genoma, produciendo un nuevo individuo que es híbrido de sus progenitores. El segundo diagrama muestra a un individuo sufriendo una mutación en la posición 4, cambiando el 0 de esa posición de su genoma por un 1.

2.5 Ejemplos específicos de AG's

Mientras el poder de la evolución gana reconocimiento cada vez más generalizado, los algoritmos genéticos se utilizan para abordar una amplia variedad de problemas en un conjunto de campos sumamente diverso, demostrando claramente su capacidad y su potencial. Esta sección analizará algunos de los usos más notables en los que han tomado parte.

2.5.1 Ingeniería eléctrica

Una matriz de puertas programable en campo (Field Programmable Gate Array, o FPGA), es un tipo especial de placa de circuito con una matriz de celdas lógicas, cada una de las cuales puede actuar como cualquier tipo de puerta lógica, interconectado con conexiones flexibles que pueden conectar celdas. Estas dos funciones se controlan por software, así que simplemente cargando un programa especial en la placa, puede alterarse al vuelo para realizar las funciones de cualquier dispositivo de hardware de la amplia variedad existente.

El Dr. Adrian Thompson ha explotado este dispositivo, en conjunción con los principios de la evolución, para producir un prototipo de circuito reconocedor de voz que puede distinguir y responder a órdenes habladas utilizando sólo 37 puertas lógicas -una tarea que se habría considerado imposible para cualquier ingeniero humano. Generó cadenas aleatorias de bits de ceros y unos y las utilizó como configuraciones de la FPGA, seleccionando los individuos más aptos de cada generación, reproduciéndolos y mutándolos aleatoriamente, intercambiando secciones de su código y pasándolo hacia la siguiente ronda de selección. Su objetivo era evolucionar un dispositivo que pudiera en principio discriminar entre tonos de frecuencias distintas (1 y 10 kilohercios), y luego distinguir entre las palabras habladas "go" (adelante) y "stop" (para).

Su objetivo se alcanzó en 3.000 generaciones, pero el éxito fue mayor de lo que había anticipado. El sistema que evolucionó utilizaba muchas menos celdas que cualquier cosa que pudiera haber diseñado un ingeniero humano, y ni siquiera necesita del componente más crítico de los sistemas diseñados por humanos -un reloj. ¿Cómo funcionaba? Thompson no tiene ni idea, aunque ha rastreado la señal de entrada a través de un complejo sistema de bucles realimentados del circuito evolucionado. De hecho, de las 37 puertas lógicas que utiliza el producto final, cinco de ellas ni siquiera están conectadas al resto del circuito de ninguna manera -pero si se les retira la alimentación eléctrica, el circuito deja de funcionar. Parece que la evolución ha explotado algún sutil efecto electromagnético de estas celdas para alcanzar su solución, pero el funcionamiento exacto de la compleja e intrincada estructura evolucionada sigue siendo un misterio (Davidson 1997[19]).

Altshuler y Linden 1997[2] utilizaron un algoritmo genético para evolucionar antenas de alambre con propiedades especificadas a priori. Los autores señalan que el diseño de tales antenas es un proceso impreciso, comenzando con las propiedades deseadas y luego determinando la forma de la antena mediante "conjeturas... intuición, experiencia, ecuaciones aproximadas o estudios empíricos" (p. 50). Esta técnica requiere mucho tiempo, a menudo no produce resultados óptimos

y tiende a funcionar bien sólo con diseños simétricos y relativamente simples. En contraste, con el método del algoritmo genético, el ingeniero especifica las propiedades electromagnéticas de la antena, y el AG sintetiza automáticamente una configuración que sirva.

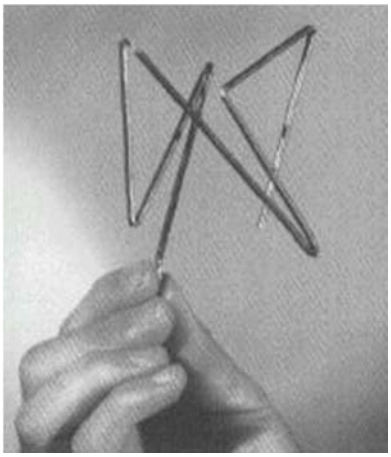

Figura 2-3 Antena genética .

Figura: Una antena genética de alambre doblado (de Altshuler y Linden 1997, figura 1).

Altshuler y Linden utilizaron su AG para diseñar una antena de siete segmentos polarizada circularmente con una cobertura hemisférica; el resultado se muestra a la izquierda. Cada individuo del AG consistía en un cromosoma binario que especificaba las coordenadas tridimensionales de cada extremo final de cada alambre. La aptitud se evaluaba simulando a cada candidato de acuerdo con un código de cableado electromagnético, y el individuo mejor de cada ronda se construía y probaba. Los autores describen la forma de esta antena, que no se parece a las antenas tradicionales y carece de una simetría obvia, como "inusualmente extraña" y "antiintuitiva" (p. 52), aunque tenía un patrón de radiación casi uniforme y con un gran ancho de banda tanto en la simulación como en la prueba experimental, adecuándose excelentemente a la especificación inicial. Los autores concluyen que un método basado en algoritmos genéticos para diseñar antenas se muestra "excepcionalmente prometedor". "... este nuevo procedimiento de diseño es capaz de encontrar antenas genéticas capaces de resolver de manera efectiva difíciles problemas de antenas, y será especialmente útil en situaciones en las que los diseños existentes no sean adecuados" (p. 52).

2.5.2 Mercados financieros

Mahfoud y Mani 1996[46] utilizaron un algoritmo genético para predecir el rendimiento futuro de 1.600 acciones ofertadas públicamente. Concretamente, al AG se le asignó la tarea de predecir el beneficio relativo de cada acción, definido como el beneficio de esa acción menos el beneficio medio de las 1.600 acciones a lo largo del periodo de tiempo en cuestión, 12 semanas (un cuarto del calendario) en el futuro. Como entrada, al AG se le proporcionaron datos históricos de cada acción en forma de una lista de 15 atributos, como la relación precio-beneficio y el ritmo de

crecimiento, medidos en varios puntos del tiempo pasado; se le pidió al AG que evolucionara un conjunto de reglas si/entonces para clasificar cada acción y proporcionar, como salida, una recomendación sobre qué hacer con respecto a la acción (comprar, vender o ninguna predicción) y un pronóstico numérico del beneficio relativo. Los resultados del AG fueron comparados con los de un sistema establecido, basado en una red neuronal, que los autores habían estado utilizando para pronosticar los precios de las acciones y administrar las carteras de valores durante tres años. Por supuesto, el mercado de valores es un sistema extremadamente ruidoso y no lineal, y ningún mecanismo predictivo puede ser correcto el 100% del tiempo; el reto consiste en encontrar un predictor que sea preciso más de la mitad de las veces.

En el experiemnto, el AG y la red neuronal hicieron pronósticos al final de la semana para cada una de las 1.600 acciones, durante doce semanas consecutivas. Doce semanas después de cada predicción, se comparó el rendimiento verdadero con el beneficio relativo predicho. Globalmente, el AG superó significativamente a la red neuronal: en una ejecución de prueba, el AG predijo correctamente la dirección de una acción el 47,6% de las veces, no hizo predicción el 45,8% de las veces y realizó una predicción incorrecta sólo un 6.6% de las veces, una precisión predictiva total de un 87,8%. Aunque la red neuronal realizó predicciones precisas más a menudo, también hizo predicciones erróneas más a menudo (de hecho, los autores especulan que la mayor capacidad del AG para no realizar predicciones cuando los datos eran dudosos fue un factor de su éxito; la red neuronal siempre produce una predicción a menos que sea restringida explícitamente por el programador). En el experimento de las 1.600 acciones, el AG produjo un beneficio relativo de un +5,47%, contra el +4,40% de la red neuronal -una diferencia estadísticamente significativa. De hecho, el AG también superó significativamente a tres índices bursátiles importantes -el S&P 500, el S&P 400 y el Russell 2000- en este periodo; la casualidad fue excluída como causa de este resultado con un margen de confianza de un 95%. Los autores atribuyen este convincente éxito a la capacidad del algoritmo genético de percatarse de relaciones no lineales difícilmente evidentes para los observadores humanos, además del hecho de que carece del "prejuicio contra las reglas antiintuitivas y contradictorias" (p. 562) de los expertos humanos.

Andreou, Georgopoulos y Likothanassis 2002[4] lograron un éxito similar utilizando algoritmos genéticos híbridos para evolucionar redes neuronales que predijeran los tipos de cambio de monedas extranjeras hasta un mes en el futuro. Al contrario que en el ejemplo anterior, donde competían AGs y redes neuronales, aquí los dos trabajaron conjuntamente: el AG evolucionó la arquitectura (número de unidades de entrada, número de unidades ocultas y la estructura de enlaces entre ellas) de la red, que luego era entrenada por un algoritmo de filtro.

Se le proporciaron al algoritmo 1.300 valores brutos diarios de cinco divisas como información histórica -el dólar estadounidense, el marco alemán, el franco francés, la libra esterlina y el dracma griego- y se le pidió que predijera sus valores futuros para los 1, 2, 5 y 20 días posteriores. El rendimiento del AG híbrido mostró, en general, un "nivel excepcional de precisión" (p. 200) en todos los casos probados, superando a otros varios métodos, incluyendo a las redes neuronales en solitario. Los autores concluyen que "se ha logrado un excepcional éxito predictivo tanto con un horizonte predictivo de un paso como de varios pasos" (p. 208) -de hecho, afirman que sus

resultados son mejores con diferencia que cualquier estrategia predictiva relacionada que se haya aplicado en esta serie de datos u otras divisas.

La utilización de los AGs en los mercados financieros ha empezado a extenderse en las empresas de corretaje bursátil del mundo real. Naik 1996[48] informa de que LBS Capital Management, una empresa estadounidense cons ede en Florida, utiliza algoritmos genéticos para escoger las acciones de los fondos de pensiones que administra. Coale 1997[17] y Begley y Beals 1995[9] informan de que First Quadrant, una empresa de inversiones de Californa que mueve más de 2.200 millones de dólares, utiliza AGs para tomar decisiones de inversión en todos sus servicios financieros. Su modelo evolucionado gana, de media, 225 dólares por cada 100 dólares invertidos durante seis años, en contraste con los 205 dólares de otros tipos de sistemas de modelos.

2.5.3 Matemáticas y algoritmia

Aunque algunas de las aplicaciones más prometedoras y las demostraciones más convincentes de la potencia de los AGs se encuentran en el campo de la ingeniería de diseño, también son relevantes en problemas "puramente" matemáticos. Haupt y Haupt 1998[34] (p. 140) describen el uso de AGs para resolver ecuaciones de derivadas parciales no lineales de alto orden, normalmente encontrando los valores para los que las ecuaciones se hacen cero, y dan como ejemplo una solución casi perfecta para los coeficientes de la ecuación de quinto orden conocida como Super Korteweg-de Vries.

Ordenar una lista de elementos es una tarea importante en la informática, y una red de ordenación es una manera eficiente de conseguirlo. Una red de ordenación es una lista fija de comparaciones realizadas en un conjunto de un tamaño dado; en cada comparación se comparan dos elementos y se intercambian si no están en orden. Koza et al. 1999[41] (p. 952) utilizaron programación genética para evolucionar redes de ordenación mínimas para conjuntos de 7 elementos (16 comparaciones), conjuntos de 8 elementos (19 comparaciones) y conjuntos de 9 elementos (25 comparaciones). Mitchell 1996[47], p. 21, describe el uso de algoritmos genéticos por W. Daniel Hillis para encontrar una red de ordenación de 61 comparaciones para un conjunto de 16 elementos, sólo un paso más allá de la más pequeña conocida. Este último ejemplo es especialmente interesante por las dos innovaciones que utiliza: cromosomas diploides y, más notablemente, coevolución de huésped/parásito. Tanto las redes de búsqueda como los casos de prueba evolucionaron conjuntamente; se les otorgó mayor aptitud a las redes de ordenación que ordenaran correctamente un mayor número de casos de prueba, mientras que se les otorgó mayor aptitud a los casos de prueba que pudieran "engañar" a un mayor número de redes de búsqueda para que ordenaran incorrectamente. El AG con coevolución rindió significativamente mejor que el mismo AG sin ella.

Un ejemplo final de AG digno de mención en el campo de la algoritmia puede encontrarse en Koza et al. 1999[41], que utilizó programación genética para descubrir una regla para el problema de clasificación por mayoría en autómatas celulares de una dimensión, una regla mejor que todas las reglas conocidas escritas por humanos. Un autómata celular de una dimensión puede imaginarse

como una cinta finita con un número dado de posiciones (celdas) en ella, cada una de las cuales puede contener el estado 0 o el estado 1. El autómata se ejecuta durante un número dado de pasos temporales; en cada paso, cada celda adquiere un nuevo valor basado en su valor anterior y el valor de sus vecinos más cercanos. (El Juego de la Vida es un autómata celular bidimensional). El problema de la clasificación por mayoría implica encontrar una tabla de reglas tal que si más de la mitad de las celdas de la cinta son 1 inicialmente, todas las celdas se ponen a 1; de lo contrario, todas las celdas se ponen a 0. El reto consiste en el hecho de que cualquier celda individual sólo tiene acceso a información acerca de sus vecinos más cercanos; por lo tanto, los conjuntos de reglas buenos deben encontrar de algún modo una manera de transmitir información sobre regiones distantes de la cinta.

Se sabe que no existe una solución perfecta a este problema -ningún conjunto de reglas puede clasificar con precisión todas las configuraciones iniciales posibles-, pero durante los últimos veinte años ha habido una larga sucesión de soluciones cada vez mejores. En 1978, tres investigadores desarrollaron la famosa regla GKL, que clasifica correctamente un 81,6% de los posibles estados iniciales. En 1993, se descubrió una regla mejor con una precisión de un 81,8%; en 1995 se encontró otra regla con una precisión de un 82,178%. Todas estas reglas requirieron para su desarrollo de un esfuerzo significativo por parte de humanos inteligentes y creativos. En contraste, la mejor regla descubierta mediante programación genética, descrito en Koza et al. 1999[41], p. 973, tiene una precisión total de 82,326% -mejor que cualquiera de las soluciones humanas desarrolladas durante las dos últimas décadas. Los autores señalan que sus nuevas reglas son cualitativamente distintas a las reglas publicadas con anterioridad, al emplear representaciones internas muy detalladas de la densidad de estados y conjuntos intrincados de señales para comunicar información a largas distancias.

Capítulo 3.
Support Vector Machine (SVM)

3.1 ¿Qué son las máquinas de vectores de soporte?

Las máquinas de Vectores de Soporte (Support Vector Machine, por sus siglas en ingles SVM) son un nuevo sistema de aprendizaje el cual ha tenido un desarrollo muy significativo en los últimos años tanto en la generación de nuevos algoritmos como en las estrategias para su implementación. SVM es un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel, en el cual las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización el cual utiliza elementos de la teoría de generalización. SVM es un sistema para entrenar máquinas de aprendizaje lineal eficientemente tanto que para clasificación como para regresión se han encontrado muchas aplicaciones como clasificación de imágenes, reconocimiento de caracteres, detección de proteínas, clasificación de patrones, identificación de funciones, etc. A continuación describiremos el sistema SVM para la identificación de función es no lineales. Para lograr esto comenzaremos dando una breve introducción a las funciones kernel, la teoría de generalización y la teoría de optimización las cuales nos ayudaran a comprender mejor al sistema SVM, y por último se describirán los métodos de SVM para regresión.

Las máquinas de vectores de soporte (SVM por sus siglas en inglés, "Support Vector Machine"), fueron desarrolladas por Vapnik (1995) , para el problema de clasificación pero la forma actual de SVM para regresión fue desarrollada en los laboratorios de AT&T por Vaptnik. SVM está ganando gran popularidad como herramienta para la identificación de sistemas no lineales, esto debido principalmente a que SVM está basado en el principio de minimización del riesgo estructural (SRM por sus siglas en inglés, "Structural Risk Minimization"), principio originado de la teoría de aprendizaje estadístico desarrollada por Vapnik, el cual ha demostrado ser superior al principio de minimización del riesgo empírico (ERM por sus siglas en ingles “Empirical Risk Minimization"), utilizado por las redes neuronales convencionales. Algunas de las razones por las que este método ha tenido éxito es que no padece de mínimos locales y el modelo solo depende de los datos con más información llamados vectores de soporte (SV por sus siglas en inglés, "Support Vectors").

Las grandes ventajas que tiene SVM son: Una excelente capacidad de generalización, debido a la minimización del riesgo estructurado. Existen pocos parámetros a ajustar; el modelo solo depende de los datos con mayor información. La estimación de los parámetros se realiza a través de la optimización de una función de costo convexa, lo cual evita la existencia de un mínimo local. La solución de SVM es sparse, esto es que la mayoría de las variables son cero en la solución de SVM, esto quiere decir que el modelo final puede ser escrito como una combinación de un número muy pequeño de vectores de entrada, llamados vectores de soporte.

SVM resuelve un problema cuadrático donde el número de coeficientes es igual al número de entradas o datos de entrenamiento. Este hecho hace que para grandes cantidades de datos las técnicas numéricas de optimización, existentes para resolver el problema cuadrático, no sean admisibles en términos computacionales. Este es un problema que impide el uso de SVM para la identificación de sistemas no lineales en línea, esto es, en casos en los que las entradas son obtenidas de manera secuencial y el aprendizaje se realiza en cada paso. En la literatura existen

algunas técnicas para la aplicación de SVM en línea. Estos intentos hicieron posible el diseño de algoritmos los cuales mejoran tanto el tiempo como el costo computacional de los algoritmos convencionales existentes para resolver el problema cuadrático. En [66] se utiliza método de optimización mínima secuencial (SMO) para dividir el problema cuadrático en una serie de problemas cuadráticos de menor dimensión los cuales pueden ser resueltos de manera analítica evitando así utilizar técnicas numéricas para el problema cuadrático mientras que en [67] se propone un algoritmo recursivo para el entrenamiento de SVM donde la idea básica consiste en encontrar las condiciones apropiadas de Kuhn- Tucker (KT) para la actualización de los coeficientes. Otra forma de lograr que la solución sea sparse es por construcción. Aquí el algoritmo comienza con una representación vacía y se agregan datos de entrenamiento de acuerdo a algún criterio. Más, sin embargo la desventaja de estas técnicas es que pueden dar una aproximación de la solución, y pueden requerir muchas operaciones por cada iteración para alcanzar un nivel adecuado de convergencia. Es indiscutible que cada día aumenta el número de aplicaciones para las cuales SVM es una herramienta muy útil. Esto debido al creciente entusiasmo, que se ha dado en las últimas décadas, por desarrollar nuevos métodos los cuales lleven al mejor desempeño del método SVM. Aun cuando SVM muestra ser un método que supera a las Redes Neuronales en cuanto a su capacidad de generalización y a la ausencia de mínimos locales, SVM sufre de otros problemas como la selección de la mejor función kernel y los problemas computacionales al realizar la identificación sobre un conjunto, muy poblado, de datos de entrenamiento.

3.2 Espacios inducidos por la función Kernel

Debido a las limitaciones computacionales de las máquinas de aprendizaje lineal estas no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real. La representación por medio del Kernel ofrece una solución alternativa a este problema, proyectando la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de las máquinas de aprendizaje lineal. La forma más común en que las máquinas de aprendizaje lineales aprenden una función objetivo es cambiando la representación de la función, esto es similar a mapear el espacio de entradas X a un nuevo espacio de características F = {ϕ(x)|x ∈ X}. Esto es:

$$x = \{x1, x2, \bullet\bullet\bullet, xn\} \longrightarrow \varphi(x) = \{\varphi(x)1, \varphi(x)2, \bullet\bullet\bullet, \varphi(x)n\}$$

Definición 6.1 Las cantidades introducidas para describir la información original o atributos son conocidas como características, mientras que a la selección de la mejor representación se le conoce como selección de características.

En la Ilustración 6.1 se muestra un mapeo de un espacio de entradas de dos dimensiones a un espacio de características de dos dimensiones, donde la información no puede ser separada por una máquina lineal en el espacio de entradas mientras que en el espacio de características esto resulta muy sencillo.

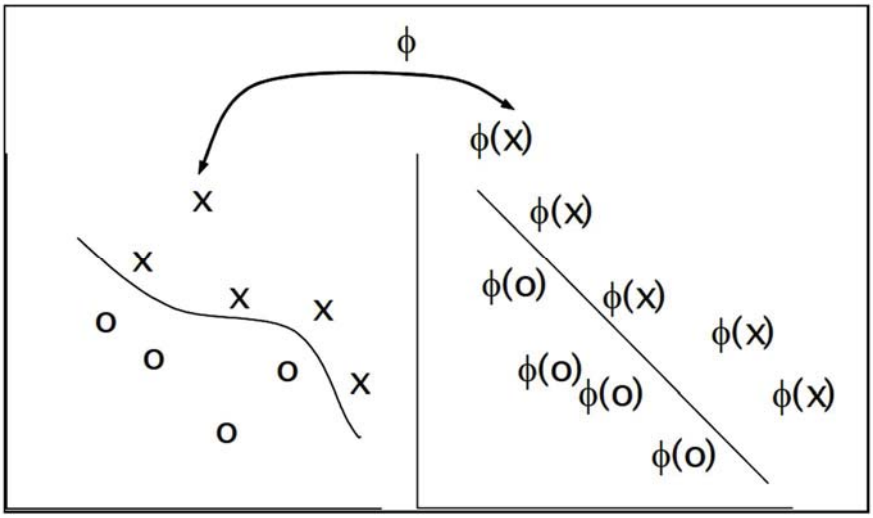

Figura 3.1 Mapeo del espacio de entradas a un espacio de características de mayor dimensión.

Las máquinas de aprendizaje lineales son funciones reales $f : X \in Rn \rightarrow Y \in R$. La función f se considera como una función lineal de x ∈ X, tal que se pude escribir como:

$$\begin{aligned} f(x) &= hw \cdot xi + b \\ &= wxT + b \\ &= \sum_{i=1}^{n} wixi + b \end{aligned}$$

Donde w es el vector de pesos y b es el bias, términos tomados de la literatura de redes neuronales. Este tipo de máquinas admiten una representación dual, esto es si definimos a $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ tenemos que la función lineal se puede escribir en su formula dual, esto es:

$$f(x) = \sum_{n=1}^{n} \alpha_i y_i \langle x_i \cdot x_i \rangle + b$$

Donde $\langle \cdot \rangle$ es el producto interno. Una propiedad importante de la representación dual es que la información de entrenamiento entra a la función a través de las entradas de la matriz de Gram $G = \langle x_i \cdot x_i \rangle$. A fin de aprender relaciones no lineales con máquinas lineales es necesario seleccionar un conjunto de características no lineales con las cuales poder rescribir la información original en una nueva representación. De ahí que el conjunto de hipótesis que se consideran son del tipo:

$$f(x) = \sum_{n=1}^{n} w_i \emptyset_i(x) + b$$

Donde $\emptyset : X \to F$ es un mapeo no lineal que va del espacio de entradas a algún espacio de características. Debido a que las máquinas de aprendizaje lineal admiten una representación dual podemos escribir la hipótesis como una combinación lineal de la información de entrada (x_i, y_i), de la siguiente manera:

$$f(x) = \sum_{n=1}^{n} \alpha_i y_i \langle \emptyset_i^T(x) \cdot \emptyset_i(x) \rangle + b$$

Esto es si podemos calcular el producto interno en el espacio de características como una función de las entradas, estaremos realizando un mapeo del espacio de entradas a el espacio de características donde se realizara el aprendizaje con una máquina lineal.

Definición 6.2 *Un Kernel K es una función, tal que para todo $x, z \in X$*

$$K(x, z) = \langle \emptyset(x) \cdot \emptyset(z) \rangle = \sum_{i=1}^{n} \emptyset_i^T(x) \emptyset_i(z)$$

Donde $\emptyset$ es un mapeo del espacio de entradas x al espacio de características F.

El uso de la función kernel hace posible realizar el mapeo de la información de entrada (x, z) al espacio de características $(\emptyset(x), \emptyset_i(z))$ de forma implícita y entrenar a la máquina lineal en dicho espacio. La única información necesaria para el entrenamiento es la matriz de Gram, dicha matriz también es conocida como la matriz kernel la cual se denota con la letra K:

$$K(x, z) = \langle \emptyset(x)_i \cdot \emptyset(x)_j \rangle_{i,j=1}^{l}$$

Una vez que se define a la matriz K, la hipótesis se puede calcular evaluando al menos l veces la matriz K de la siguiente manera:

$$f(x) = \sum_{n=1}^{\mathrm{l}} \alpha_i y_i K(x_i, x) + b$$

Comentario 6.1 Utilizando la función kernel no es necesario calcular explícitamente el mapeo $\emptyset : X \to F$ para aprender en el espacio de características.

Algunas de las propiedades que debe cumplir la función kernel para definir un espacio de características son las siguientes:
1.- Simétrica:

$$K(x,z) = \langle \emptyset(x) \cdot \emptyset(z) \rangle = \langle \emptyset(z) \cdot \emptyset(x) \rangle = \sum_{i=1}^{l} \emptyset_i(x)\emptyset_i(z) = K(z,x)$$

2.- Desigualdad de Cauchy-Schwarz:

$$K(x,z)^2 = \langle \emptyset(x) \cdot \emptyset(z) \rangle^2 \leq \|\emptyset(x)\|^2 \|\emptyset(z)\|^2 = \langle \emptyset(x) \cdot \emptyset(z) \rangle \langle \emptyset(x) \cdot \emptyset(z) \rangle = K(x,x)K(z,z)$$

Estas condiciones, como sea, no son suficientes para la existencia de un espacio de características. En esta sección se introduce el teorema de Mercer el cual provee las condiciones que debe cumplir una función K para ser un kernel el cual induzca un espacio de características.

Proposición 6.1 *Sea X un espacio de entradas finito con $K(x,z)$ una función simétrica sobre X. Entonces K(x,z) es un kernel si y solo si la matriz*

$$K = (K\ (x_i, x_j))_{i,j=1}^{n}$$

Es positiva semidefinida.

En un espacio de Hilbert también se puede definir un kernel, puesto que el espacio de Hilbert es un espacio con producto interno que satisface las propiedades lineales. Introduciendo un elemento de ponderación λ se define el kernel de la siguiente forma:

$$\langle \phi(x) \cdot \phi(z) \rangle = \sum_{i=1}^{\infty} \lambda_i \emptyset_i(x)\emptyset_i(z) = k(x,z)$$

El teorema de Mercer da las condiciones necesarias y suficientes para que una función simétrica y continua $k(x,z)$ admita dicha representación, con $\lambda_i \geq 0$. Esto es equivalente a que $k(x,z)$ sea un producto interno en el espacio de características $F \supseteq \emptyset(X)$, don de F es un espacio l_2 de secuancias $\psi = (\psi_1, \psi_2, \cdots, \psi_{i,}, \cdots)$ para el cual $\sum_{i=1}^{\infty} \lambda_i \psi_i^2 < \infty$. Esto induce implícitamente un espacio de características en el cual se puede representar una función lineal

$$f(x) = \sum_{i=1}^{\infty} \lambda_i \psi_i \emptyset_i(x) + b = \sum_{j=1}^{l} \alpha_j y_j K(x, x_j) + b$$

Con $\psi = \sum_{j=1}^{l} \alpha_i y_j \phi(x_j)$. Note que en la segunda representación el número de términos en la sumatoria es igual al número de ejemplos de entrenamiento.

Teorema 6.1 *(Mercer) Sea X un subconjunto compacto de $\mathbb{R}^n$. Sea K una función simetrica continua tal que el operador $T_K : L_2(X) \to L_2(X)$*

$$(T_K f)(\cdot) = \int_X K(\cdot, x) f(x) dx$$

Es positivo, esto es

$$\int_{X\times X} K(x,z)f(x)f(z)dxdz \geq 0$$

Para todo $f \in L_2(X)$. Entonces podemos expandir $K(x,z)$ en una serie uniformemente convergente en términos de las eigenfunciones de T_K , $\emptyset \in L_2(X)$, con $\|\emptyset i\|_{L_2} = 1$ y los eigenvalores asociados $\lambda_i \geq 0$.

$$K(x,z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z)$$

Comentario 6.2 *La condición de positividad es equivalente a requerir que para cualquier subconjunto finito de X, su matriz correspondiente sea positiva semidefinida*

$$K(x,z) = \int_{X\times X} K(x,z)f(x)f(z)dxdz \geq 0 \in L_2$$

Esto es para verificar que una función simétrica y continua sea un kernel es verificar la Observación 6.2. Esto es que la matriz definida al restringir la función a un conjunto finito sea definida positiva.

Proposición 6.2 *Sean K_1 y K_2 kernels sobre $X \times X, X \subseteq \mathbb{R}^n, a \in \mathbb{R}^+$, con f una función real sobre X*

$$\emptyset: X \to \mathbb{R}^m$$

Con K_3 un kernel sobre $\mathbb{R}^m \times \mathbb{R}^m, B^{n\times n}$ una matriz semidefinida positiva, entonces las siguientes funciones son kernels:

1.- $K(x,z) = K_1(x,z) + K_2(x,z)$
2.- $K(x,z) = aK_1(x,z), a \in \mathbb{R}$
3.- $K(x,z) = K_1(x,z)K_2(x,z)$
4.- $K(x,z) = f(x)f(z)$
5.- $K(x,z) = K_3(\emptyset(x), \emptyset(z))$
6.- $K(x,z) = x^T Bz$

A continuación se dan algunas funciones que satisfacen las condiciones de Mercer.
Lineal: Este kernel es una transformación lineal del tipo

$$K(x,z) = \langle Ax \cdot Az\rangle = x^T A^T Az = x^T Bz$$

Donde $B = A^T, A$ es una matriz semidefinida positiva.
Polinomial: El mapeo polinomial es un método muy popular para modelar funciones no lineales,

$$K(x,z) = \langle x,x\rangle^d$$
$$K(x,z) = (\langle x,x\rangle + c)^d$$

Con $c \in \mathbb{R}$. En la práctica se prefiere utilizar el segundo kernel debido a que se evitan problemas de que el Hesiano se vuelva cero.

Funciones de base Radial:

$$K(x,z) = exp\left(-\|x-z\|^2 / \sigma^2\right)$$

Las funciones de base radial (RBF) son también conocidas como funciones Gaussianas.

3.3 SVM para Regresión

En SVM la meta es encontrar una función $f(x)$ que tenga a lo más una desviación ε de la salida y_i para todos los datos de entrenamiento, y al mismo tiempo, que sea lo más mínima posible. En otras palabras, no nos preocupamos por errores menores a ε, pero si de aquellos que sean mayores. Considere el problema de aproximar un conjunto de entrenamiento $\{x_i, y_i\}, x \in \mathbb{R}^n, y \in \mathbb{R}$, por medio de una función lineal $f: X \subseteq \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} f(x) &= \langle w \cdot x \rangle + b \\ &= x^T w + b \\ &= \sum_{i=1}^{n} w_i x_i + b \end{aligned}$$

Donde $\langle \cdot \rangle$ es el símbolo del producto interno, y $x = \{x_1, x_2, \cdots, x_n\}$, $y = \{y_1, y_2, \cdots, y_n\}$ y $w = \{w_1, w_2, \cdots, w_n\}$. Donde w es el vector de pesos y b es el bias. En caso de la formula anterior se busca encontrar w lo más mínima posible. Para asegurar esto, una forma es minimizar la norma Euclidiana, p.e. $\|w\|^2$. Formalmente podemos escribir este problema como un problema de optimización convexo: fue que

$$\text{Minimizar } \frac{1}{2}\|w\|^2$$

Sujeto a:

$$y_i - (\langle w \cdot x_i \rangle + b) \leq \varepsilon$$
$$y_i - (\langle w \cdot x_i \rangle + b) \leq -\varepsilon$$

La sunción clave en la formula fue que la función $f(X)$ Existe y aprozima a todos los pares (x_i, y_i) con una precisión ε, o en otras palabras que el problema convexo es factible. Algunas veces esto no es el caso por o cual se introducen variables de perdida. En SVM para regresión la idea básica consiste en realizar un mapeo de los datos de entrenamiento $x \in X$, a un espacio de mayor dimensión F a través de un mapeo no lineal $\emptyset: X \to F$, donde podemos realizar una regresión lineal. En la figura a continuación se puede observar la arquitectura generada por SVM para regresión, también se puede observar que el regresor final solo depende de todos aquellos datos que contienen la mayor información posible para generar el regresor, llamados vectores de soporte.

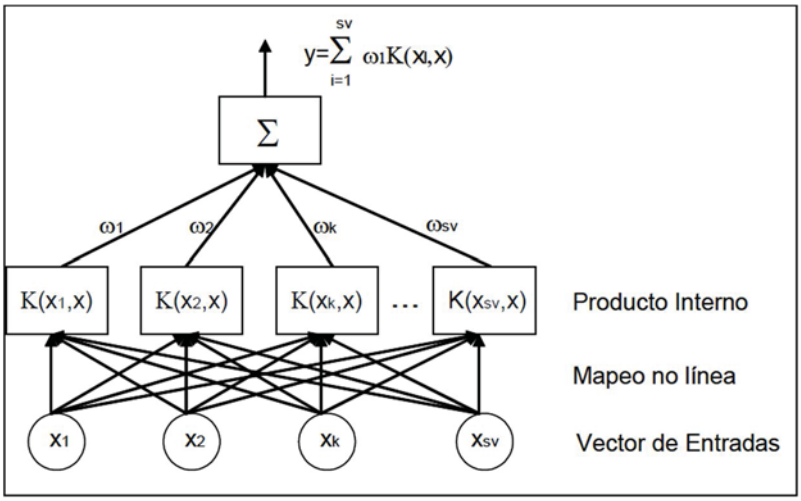

Figura 3.2 Arquitectura de SVM para regresión

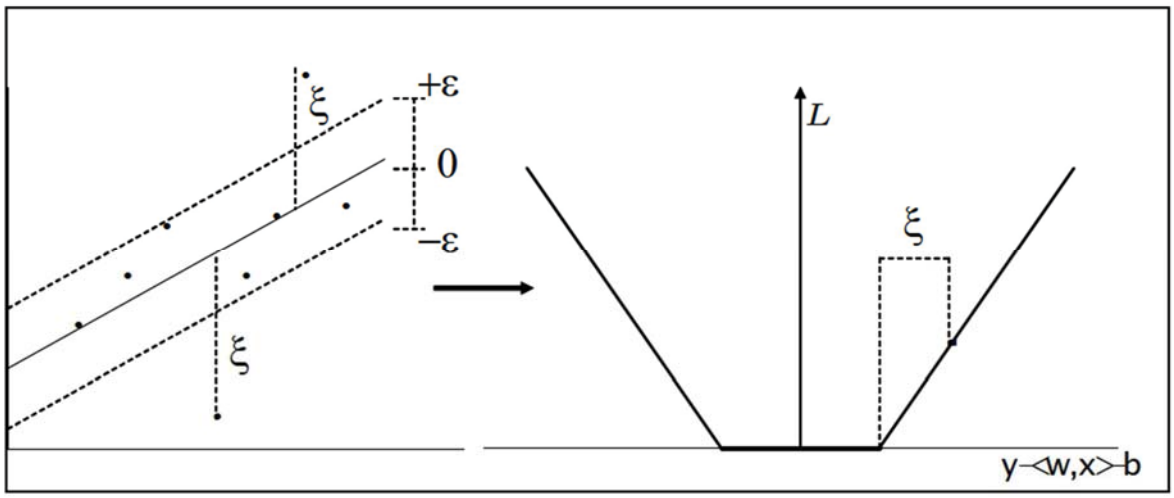

Figura 3.3 Variables de Perdida ξ.

En la Figura 7.2 se muestra las variables de perdida para un ejemplo de identificación con banda ε insensitive.

SVM utiliza una función de costo originalmente propuesta por Vaptnik(1995) se define a continuación la función de costo $\varepsilon -$ insensitive.

Definición: *La función de costo, o discrepancia, $\varepsilon -$insensitive, $L^{\varepsilon}(x, y, f)$ se describe commo*

$$L^{\varepsilon}(x, y, f) = |y - f(x)| = máx(0, |y - f(x)| - \epsilon)$$

Donde f es una función real con dominio $X, x \in X, y \in \mathbb{R}$.

3.4 Mínimos cuadrados – máquina de vector de soporte (LS-SVM)

El método de Mínimos cuadrados SVM para identificación trabaja fuera de línea pero la ventaja que tiene en comparación con los métodos ε insensitive y cuadrático es que la solución del problema se realiza a través de un conjunto de ecuaciones. Este método puede tratar una cantidad más considerable de datos de entrenamiento dado un conjunto de entrenamiento $\{x_i, y_i\}_{i=1}^{n}$, $x \in \mathbb{R}^n, y \in \mathbb{R}$. En el espacio de características el modelo dado por las Maquinas de Vectores de Soporte toma la siguiente forma:

$$f(x) = \sum_{i=1}^{n} w_i \emptyset_i(x) + b$$

Donde el mapeo $\emptyset: X \to F$, mapea el espacio de entradas a algún espacio de características de mayor dimensión. En el método LS-SVM se plantea el siguiente problema de minimización

$$\text{Mín } J(w, \xi) = \frac{1}{2}\|w\|^2 + \frac{c}{2}\sum_{i=1}^{l} \xi_i^2$$
$$\text{Sujeto a: } y_i - \langle w \cdot \phi(x_i)\rangle = \xi_i$$

Este problema corresponde al problema de SVM Rifge regresión. El Lagrangiando está dado por:

$$L(w, b, \alpha, \xi) = \frac{1}{2}\|w\|^2 \frac{c}{2}\sum_{i=1}^{l} \xi_i^2 - \sum_{i=1}^{l} \alpha_i(\langle w \cdot \phi(x_i)\rangle + b + \xi_i - y_i)$$

Donde α_i son los multiplicadores de Lagrange. Calculando las derivadas parciales del Lagrangiano en la función, con respecto de todas las variables e igualando a cero podemos obtener las condiciones de optimización, esto debido a que la función es convexa

$$\frac{\partial L}{\partial w} = 0 \ \to w = \sum_{i=1}^{l} \alpha_i \emptyset(x_i)$$
$$\frac{\partial L}{\partial w} = 0 \ \to \sum_{i=1}^{l} \alpha_i = 0$$
$$\frac{\partial L}{\partial w} = 0 \ \to \alpha_i = C\xi_i$$
$$\frac{\partial L}{\partial w} = 0 \ \to y_i = \langle w \cdot \phi(x_i)\rangle + b + \xi_i$$

Eliminando w y ξ y utilizando la función kernel, obtenemos el siguiente sistema de ecuaciones lineales:

$$y_i = \sum_{i=1}^{l} \alpha_i \phi(x_i)\phi(x_i) + b + \frac{\alpha_i}{C}$$
$$\sum_{i=1}^{l} \alpha_i = 0$$
$$\begin{bmatrix} 0 & \bar{1}^T \\ \bar{1} & K + \frac{1}{C} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}$$

Donde $y = [y_1, \cdots, y_i]^T$. Entonces el vector $1 = [1; \cdots 1]^T$. Dado este sistem de ecuaciones, la fórmula de matrices antes mencionada, el modelo utilizado por el método LS-SVM se expresa de la siguiente forma:

$$\hat{f}(x) = K(x, x_i)\alpha + b$$

Nótese que este método no requiere de la determinación del parámetro de precisión ϵ el cual esta relacionado con la función de costo $\epsilon -$ insensitive del método propuesto por Vaptnik. La gran desventaja de este método es que la solución del modelo no es sparse, esto es que ninguna de las soluciones α_i se desvanece en la solución óptima, por ende el modelo depende de todos los datos de entrenamiento.

Capítulo 4. Técnicas para determinar el pronóstico. (AGSVM)

La versión de regresión de las máquinas de vector de soporte, son una técnica alternativa y muy poderosa para resolver problemas de regresión, introduciendo la alternativa de la función de pérdida. De aquí en adelante será mencionado como SVR.

La formulación SVR sigue el principio de minimización del riesgo estructural, tratando de minimizar una cota superior del error de generalización en lugar de minimizar el error de predicción en el conjunto de entrenamiento (principio de minimización del riesgo empírico). Esto dota al SVR con un mayor potencial para generalizar la relación de entrada-salida aprendiendo durante su fase de formulación para hacer predicciones buenas para los datos de entrada.

Para lograr determinar con éxito el pronóstico deseado, es necesario primero establecer ciertas premisas, tales como que se utilizará un soporte de vector para regresión (SVR), dentro de éste el encargado de proporcionarle los parámetros óptimos será el algoritmo genético, de aquí en adelante AGSVM.

Para construir un modelo de AGSVM eficiente, primero se deben buscar los parámetros con mucho cuidado, y como ya se ha explicado en este documento, el algoritmo genético es el que tiene las características esenciales para realizar este trabajo. Es decir que el algoritmo genético será el que abastezca al SVR de los valores óptimos.

Los parámetros del AGSVM incluyen:

- Función Kernel: Esta función es usada para construir una decisión no linear que traspasa los planos a una dimensión mayor en las entradas del SVR. Para esto usaremos una ***función Gaussiana*** que entrega un rendimiento de predicción mucho mejor, según [69].
- Parámetro de regularización C: C determina el costo de intercambio entre la minimización del error de entrenamiento y la minimización de la complejidad del modelo.
- Ancho de banda de la función Kernel (σ^2): σ^2 representa la varianza de la función Kernel Gaussiana.

Generalmente el proceso de selección de los parámetros se ha realizado con procesos de prueba y error, lo que conlleva a una demora sustancial en encontrar los resultados deseados para los parámetros. Además de ser resultados poco óptimos tiene un costo de tiempo altísimo para llegar algún valor de peso.

En contraste a esto, en este trabajo se realizará el proceso de selección de los parámetros en base a los algoritmos genéticos, con el fin de lograr una búsqueda óptima en los parámetros y ayudar a la predicción eficiente.

El modelo de GA-SVR, optimiza de los valores de los parámetros de SVR dinámicamente, a través de un proceso evolutivo en el algoritmo genético y utilizando parámetros de adquisición para la construcción del modelo SVR a fin de proceder al pronóstico.

Para detallar mejor el flujo que el modelo planteado llevará a cabo en la ejecución del simulador se presenta el siguiente diagrama que refleja la idea:

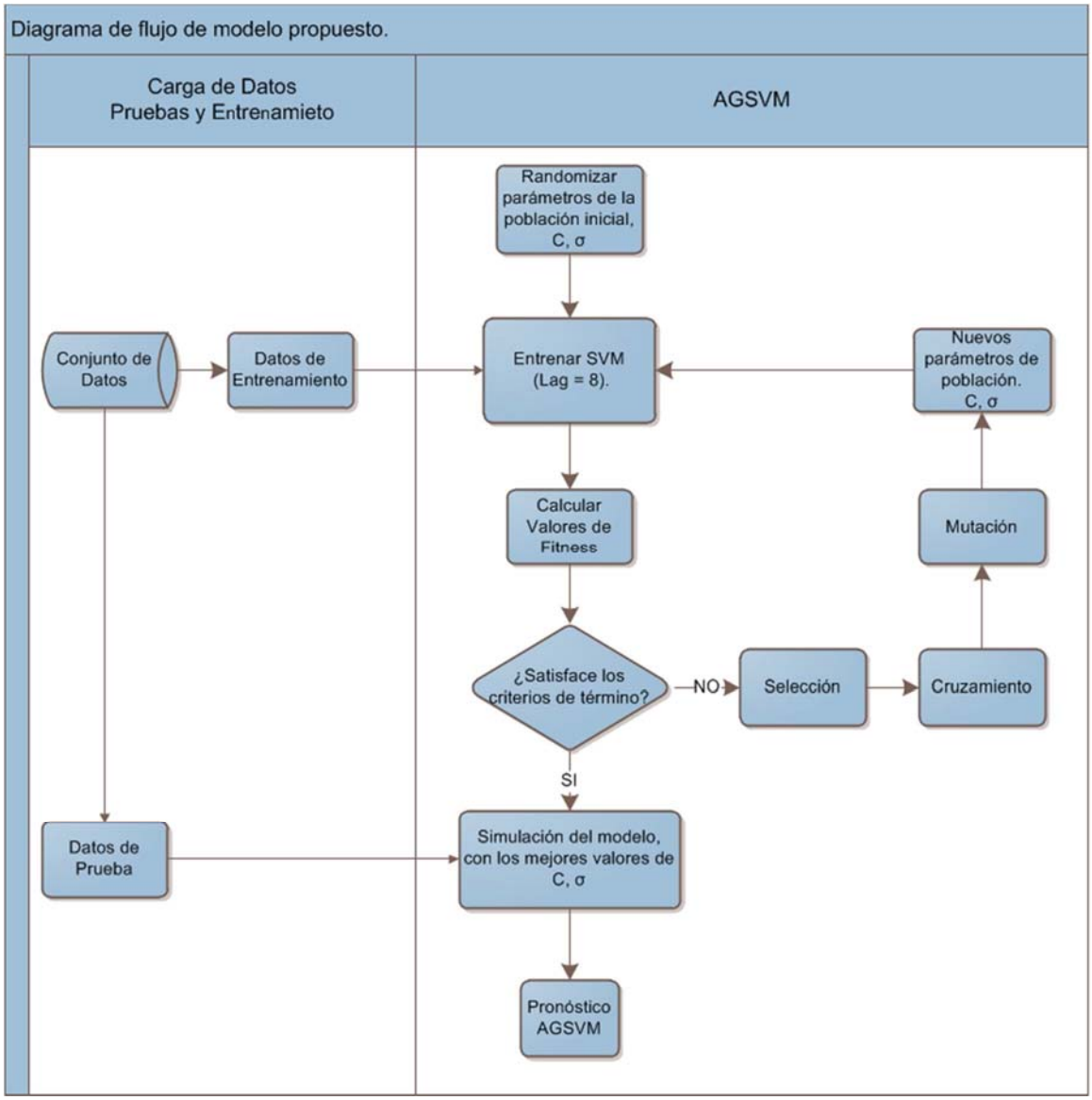

Figura 4-1 Modelo de Algoritmo genético con vector de soporte para regresión.

La población inicial consiste en cromosomas generados para el AG de manera aleatoria para encontrar valores óptimos en los parámetros del AGSVM. Estos valores de los dos parámetros, por ejemplo C y σ^2 serán codificados directamente en los cromosomas con los datos de valores reales. Los detalles serán explicados a continuación:

1) *Representación*: Cuando el AG resuelve problemas de optimización, los valores relativos de los valores de los parámetros pueden ser usados desde un cromosoma, distinto al tradicional AG binario, que debe ser pasado a una codificación binaria. Los dos parámetros del AGSVM, C y σ^2 son directamente asignados para generar el cromosoma en el método propuesto. El cromosoma X será representado como X = *p1*, *p2*, donde *p1* y *p2* denotan la regularización de los parámetros C y σ^2 respectivamente.
2) *Definición de entrenamiento*: el entrenamiento de los datos es fácil de calcular, pero es propenso a ser sobre ajustado. Para solucionar este problema se puede utilizar una técnica de validación cruzada. En este contexto, el cruzamiento de *n-pliegues*

entrega un mejor comportamiento entre el costo computacional y la estimación de parámetros fiables, esto ha sido adoptado satisfactoriamente por Duan [70]. En la validación del cruzamiento de *n-pliegues* los datos de entrenamientos son aleatoriamente divididos en *n* subconjuntos mutuamente exclusivos (pliegues), de tamaños aproximadamente iguales. La función de regresión es construida con el conjunto de parámetros entregada (C y σ^2), usando *n*-1 subconjuntos de datos de entrenamiento. El rendimiento del conjunto de parámetros es medido por MAPE (Mean absolute percentage error, en español, porcentaje de error medio absoluto) del último sub conjunto. Este proceso se repite *n* veces hasta que cada uno de ellos haya sido testeado. Promediando el MAPE con los *n* ensayos ($MAPE_{\ validacion\ de\ cruce}$) entrega un estimación esperada generalizada del error esperado de los datos de entrenamientos de tamaño $(n - 1/n) \cdot l$, donde l es la cantidad de datos de entrenamiento. Finalmente, conjunto de datos con mejor performance se especifica. Convencionalmente el error de entrenamiento de la validación de cruce en *n*-pliegues es aplicado para estimar el error generalizado (*n* = 5, sugerido por Duan [70]).Por lo tanto, la función de ajuste es definida como $MAPE_{\ validacion\ de\ cruce}$en un método de entrenamiento de 5 pliegues de validación de cruce como sigue:

$$Min\ f = MAPE_{\ validacion\ de\ cruce},$$

$$MAPE_{\ validacion\ de\ cruce} = \frac{\sum_{i=1}^{l} |a_i - p_i| / a_l}{l} \times 100\%$$

Donde l es el número de muestra de datos de entrenamiento; a_i es el valor actual, y p_i el valor pronosticado. La solución con un $MAPE_{\ validacion\ de\ cruce}$ pequeño de los datos de entrenamiento tiene un menor valor de ajuste, y esto permite tener una mejor chance de hacer sobrevivir las generaciones sucesivas.

3) *Inicialización de la población*: en este estudio seleccionaremos, de manera aleatoria, 10, 20, y 30 cromosomas, tema que será modificado según el tiempo de convergencia y la diversificación de la población en estudio.
4) *Evaluación de ajuste*: Para cada cromosoma, el valor de ajuste será calculado con las ecuaciones de $Min\ f = MAPE_{\ validacion\ de\ cruce}$.
5) *Selección*: Se empleará una ruleta estándar para seleccionar la cantidad de cromosomas acorde a la relación de la población actual.
6) *Cruzamiento*: La simulación de cruzamiento binario [71,72] será aplicado aleatoriamente en pares de cromosomas. La probabilidad de crear un cromosoma nuevo por cada par será establecido en 0,5. Los nuevos cromosomas serán considerados en la nueva población.
7) *Mutación*: La operación de mutación seguirá la operación de cruzamiento y determinará qué cromosoma será mutado para la siguiente generación de población. Para seleccionar estos cromosomas se utilizará una probabilidad de 0,1.
8) *Estrategia elitista*: El valor de ajuste será calculado con los cromosomas de la nueva población, si el valor de ajuste de la nueva población es menor que la población antigua, entonces los cromosomas antiguos podrán ser reemplazados por los nuevos cromosomas con mínimo valor de ajuste.
9) *Criterio de término*: el proceso será repetido de 1 hasta 10 hasta que el número de generaciones sea igual a 100, 200 y 300 respectivamente, o que el proceso de optimización de los parámetros no obtenga mejores resultados (el promedio del valor de variación ajuste no sea mayor a 10e-900).

Capítulo 5. Discusión de Resultados.

Hecho las simulaciones pertinentes, se han encontrado diversos resultados, según las poblaciones aleatorias generadas. Para para detallar estos resultados, se plasmarán en un cuadro que muestra los distintos resultados obtenidos para cada número de iteraciones, y tamaño de población.

5.1 Representación de los resultados

A continuación se mostrarán los mejores resultados obtenidos para cada caso, tamaño de población P = {10, 20, 30}, con iteraciones I = {1..100, 1..200, 1..300} para cada P.

P=10/ I=100

Nº Sim	MSE	R2	MAPE	C	Sig
1	0.00305281	0.998041373	1.750439072	3.01471054	0.07641993
2	0.005819312	0.904087754	2.302046314	1.44362659	0.24620217
3	0.000866675	0.99983644	0.924541572	7.95753426	0.05811057
4	0.003761768	0.997279348	1.938890595	2.50652036	0.0771686
5	3.17E-07	0.999998859	0.01709587	565.091326	0.12536778
6	1.95E-06	0.999965109	0.032002543	272.418486	0.21214707
7	0.009866454	0.997539459	2.944969697	0.60623872	0.04924539
8	0.009498328	0.982025755	2.993245735	0.65903404	0.07920109
9	7.72E-03	0.999705769	2.591059139	0.65903404	0.07920109
10	0.009666143	0.806589837	3.069778651	0.59356335	0.20265553
Promedio	**0.005025823**	**0.96850697**	**1.856406919**	**85.4950074**	**0.12057192**
STA	**0.004029944**	**0.064143088**	**1.168373654**	**188.717438**	**0.07239929**
Min	**3.17E-07**	**8.07E-01**	**1.71E-02**	**5.94E-01**	**4.92E-02**
Max	**9.87E-03**	**1.00E+00**	**3.07E+00**	**5.65E+02**	**2.46E-01**

Tabla 5-1 Métricas para P = 10, I = 1 a 100.

P = 10/ I = 200

Nº Sim	MSE	R2	MAPE	C	Sig
1	0.005558885	0.995181823	2.337021107	2.47311159	0.07594728
2	0.011720514	0.99877268	3.105048854	0.35621937	0.04091532
3	0.007404166	0.63338445	3.194614793	4.70970736	0.01164583
4	0.00361965	0.999755497	1.830881861	2.62414595	0.04379806
5	0.004928177	0.94782536	2.170007655	1.8066722	0.20421873
6	0.008686435	0.999390263	2.748131375	0.81162385	0.04049348
7	0.003484165	0.999787209	1.795022153	2.7231701	0.04305711
8	0.000167306	0.999999971	0.376878186	21.4149898	0.01989909
9	0.005566523	0.927565967	2.29129643	1.54537834	0.21858858
10	0.011405585	0.966775586	3.220953176	0.38662935	0.0797665
Promedio	**0.006254141**	**0.94684388**	**2.306985559**	**3.88516479**	**0.077833**
STA	**0.003621456**	**0.113161229**	**0.862086879**	**6.29676744**	**0.07352608**
Min	**1.67E-04**	**6.33E-01**	**3.77E-01**	**3.56E-01**	**1.16E-02**
Max	**1.17E-02**	**1.00E+00**	**3.22E+00**	**2.14E+01**	**2.19E-01**

Tabla 5-2 Métricas para P = 10, I = 1 a 200.

P=10/ I=300

Nº Sim	MSE	R2	MAPE	C	Sig
1	0.004519269	0.966775586	0.966775586	0.38662935	0.0797665
2	0.005862869	0.972015462	2.439263239	1.4859215	0.13391403
3	0.008116327	1	2.374264178	0.96576623	0.00717005
4	2.30E-05	0.99897899	0.115712123	112.688906	0.02405476
5	1.77E-07	0.999999999	0.012747374	730.982376	0.03157974
6	0.004248677	0.99575806	2.06528542	2.20770218	0.08350634
7	0.001695996	0.996262311	1.375607286	5.10799195	0.00479284
8	0.007861502	0.987565224	2.758494573	0.96825271	0.08177339
9	0.005240292	0.935076566	2.221026184	1.67005913	0.21741518
10	0.003135191	0.998787243	1.757650752	2.99161798	0.06601318
Promedio	**0.004070329**	**0.985121944**	**1.608682671**	**85.9455223**	**0.0729986**
STA	**0.002887087**	**0.021285321**	**0.968735031**	**229.301991**	**0.06493184**
Min	**1.77E-07**	**9.35E-01**	**1.27E-02**	**3.87E-01**	**4.79E-03**
Max	**8.12E-03**	**1.00E+00**	**2.76E+00**	**7.31E+02**	**2.17E-01**

Tabla 5-3 Métricas para P = 10, I = 1 a 300.

P=20/ I=100 **I**

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	4.34E-03	0.999193453	2.026748665	2.1776702	0.05332311
2	3.70E-03	0.999132731	1.885726166	2.56356597	0.0571992
3	4.14E-03	0.985810391	2.058371792	2.23739598	0.12937934
4	7.73E-03	1	2.274434095	1.05979724	0.00399811
5	1.87E-03	0.999999999	1.198696252	4.68461676	0.01205367
6	4.23E-03	0.983310265	2.077330042	2.18362768	0.13622198
7	2.03E-03	1	1.192324853	4.43047328	0.00562496
8	1.50E-02	0.985842427	3.231503256	2.25672801	0.05331314
9	4.15E-03	0.997651545	2.02359337	2.25672801	0.05331314
10	3.98E-03	0.997791511	1.983272396	2.37022011	0.07091176
Promedio	**0.00511626**	**0.994873232**	**1.995200089**	**2.62208233**	**0.05753384**
STA	**0.003813161**	**0.006906414**	**0.568634666**	**1.09730049**	**0.04620617**
Min	**1.87E-03**	**9.83E-01**	**1.19E+00**	**1.06E+00**	**4.00E-03**
Max	**1.50E-02**	**1.00E+00**	**3.23E+00**	**4.68E+00**	**1.36E-01**

Tabla 5-4 Métricas para P = 20, I = 1 a 100.

P= 20/ I=200

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	0.999973318	1.61093383	3.205781738	0.03157924	0.05332311
2	0.998521568	1.992154826	2.314482451	0.06303295	0.0571992
3	0.966802416	2.406948121	1.525513906	0.14642271	0.12937934
4	0.99670358	2.367183501	1.537347498	0.06699759	0.00399811
5	0.997079183	1.971469095	2.42292039	0.07765633	0.01205367
6	0.999521008	1.818327925	2.715066291	0.05081644	0.13622198
7	0.999916791	1.745429128	2.813769875	0.03670873	0.00562496
8	1	1.164400635	4.77049165	0.00910921	0.05331314
9	0.999976702	1.647143377	3.065591108	0.03070796	0.05331314
10	1	1.145170203	4.851863902	0.00823165	0.07091176
Promedio	**0.995849457**	**1.786916064**	**2.922282881**	**0.05212628**	**0.05753384**
STA	**0.010282836**	**0.428162748**	**1.144191857**	**0.04052607**	**0.04620617**
Min	**9.67E-01**	**1.15E+00**	**1.53E+00**	**8.23E-03**	**4.00E-03**
Max	**1.00E+00**	**2.41E+00**	**4.85E+00**	**1.46E-01**	**1.36E-01**

Tabla 5-5 Métricas para P = 20, I = 1 a 200.

P=20/I=300

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	1.97E-03	0.999999995	1.244755125	4.48363266	0.01391721
2	3.78E-03	0.996874381	1.946696598	2.49488464	0.08032581
3	3.22E-03	0.999961598	1.69174425	2.94143132	0.03278212
4	3.68E-03	0.999327927	1.87344911	2.57793568	0.05395633
5	3.59E-03	0.999978448	1.7696944	2.65574603	0.02962202
6	3.77E-03	0.99915008	1.903344244	2.51132891	0.05652416
7	5.89E-03	0.807045238	3.313092472	2.64643141	0.02787086
8	2.05E-03	0.999999977	1.278394669	4.35158611	0.01556004
9	2.11E-03	0.999999948	1.304735898	4.24275089	0.01659715
10	1.48E-03	1	1.024937539	5.61596514	0.00636471
Promedio	**0.003153551**	**0.980233759**	**1.73508443**	**3.45216928**	**0.03335204**
STA	**0.00130202**	**0.060859988**	**0.643248552**	**1.12002553**	**0.02337047**
Min	**1.48E-03**	**8.07E-01**	**1.02E+00**	**2.49E+00**	**6.36E-03**
Max	**5.89E-03**	**1.00E+00**	**3.31E+00**	**5.62E+00**	**8.03E-02**

Tabla 5-6 Métricas para P = 20, I = 1 a 300.

P=30/I=100

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	2.95E-03	0.999993822	1.592561138	3.18871446	0.02593644
2	3.82E-03	0.999241343	1.945954011	2.48461359	0.02342394
3	1.82E-03	1	1.095160147	4.85080053	0.0021215
4	1.83E-03	1	1.154863986	4.78995272	0.00824126
5	8.88E-03	0.999403902	2.772254131	0.77636659	0.04008106
6	4.29E-03	0.999999999	1.799472382	2.2428832	0.01191074
7	1.83E-03	1	1.149422078	4.79097121	0.00755427
8	3.74E-03	0.997941868	1.924207508	2.52085307	0.07136083
9	1.82E-03	1	1.094433416	4.85227392	0.00206592
10	3.00E-03	0.999986176	1.618966148	3.13617772	0.02868575
Promedio	**0.003397255**	**0.999656711**	**1.614729495**	**3.3633607**	**0.02213817**
STA	**0.002144054**	**0.000665712**	**0.532148321**	**1.41479145**	**0.02138989**
Min	**1.82E-03**	**9.98E-01**	**1.09E+00**	**7.76E-01**	**2.07E-03**
Max	**8.88E-03**	**1.00E+00**	**2.77E+00**	**4.85E+00**	**7.14E-02**

Tabla 5-7 Métricas para P = 30, I = 1 a 100.

P=30/I=200

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	2.64E-03	0.999997635	1.497264282	3.51457552	0.02349401
2	3.63E-03	0.999732562	1.834863631	2.61892481	0.04454759
3	3.17E-03	0.999862745	1.707100537	2.97634678	0.04058031
4	3.35E-03	0.999986074	1.706007185	2.83471196	0.02823536
5	6.40E-03	0.999544207	2.39792746	1.35478635	0.04217363
6	2.08E-03	0.999999962	1.293472281	1.35478635	0.04217363
7	3.76E-03	0.997289948	1.938583085	2.50708486	0.07708597
8	4.14E-03	0.997862826	2.017234982	2.2789419	0.06915758
9	3.42E-03	0.999984356	1.724507864	2.78072504	0.02858427
10	3.71E-03	0.998937064	1.895638728	2.5521555	0.06004436
Promedio	**0.003630349**	**0.999319738**	**1.801260004**	**2.47730391**	**0.04560767**
STA	**0.001138688**	**0.00098391**	**0.299192095**	**0.6774762**	**0.01790249**
Min	**2.08E-03**	**9.97E-01**	**1.29E+00**	**1.35E+00**	**2.35E-02**
Max	**6.40E-03**	**1.00E+00**	**2.40E+00**	**3.51E+00**	**7.71E-02**

Tabla 5-8 Métricas para P = 30, I = 1 a 200.

P=30/I=300

Nº Simulacion	MSE	R2	MAPE	C	Sig
1	2.94E-03	0.999989932	1.596590475	3.20069989	0.02760409
2	0.004711713	0.995978338	2.16317757	1.97446687	0.07816847
3	0.003527941	0.995651314	1.895101511	2.6584109	0.09212192
4	0.003750509	0.997890458	1.927851266	2.51772879	0.0717665
5	3.27E-03	1	1.565539383	2.93920351	0.01086335
6	3.89E-03	0.999136282	1.930027342	2.46721727	0.05617831
7	3.93E-03	0.999940958	1.867243798	2.42564756	0.0337213
8	2.30E-03	1	1	4.03554896	0.00292758
9	2.38E-03	0.999999924	1.385380831	3.85336273	0.01696333
10	0.003770281	0.995706828	1.954626308	2.49271976	0.08882107
Promedio	**0.00344532**	**0.998429403**	**1.752457961**	**2.85650062**	**0.04791359**
STA	**0.000744074**	**0.001945027**	**0.29115765**	**0.65918332**	**0.03358025**
Min	**2.30E-03**	**9.96E-01**	**1.24E+00**	**1.97E+00**	**2.93E-03**
Max	**4.71E-03**	**1.00E+00**	**2.16E+00**	**4.04E+00**	**9.21E-02**

Tabla 5-9 Métricas para P = 30, I = 1 a 300.

Como se muestran los resultados, en las distintas instancias de pruebas, el parámetro de medición del error MSE se mantiene en promedio en un número bajo el rango de 10e-3, con un un R2 siempre sobre el 98%, salvo en un par de ocasiones. Esto nos lleva a considerar que los resultados del modelo, son de bajo error, y de un buen pronóstico.

5.1.1 Peor resultado en simulación:

Población = 10, iteraciones = 200, MSE= 1.17E-02, R^2=0.998, MAPE=3.105, C= 21.4150, Sig^2=0.0199

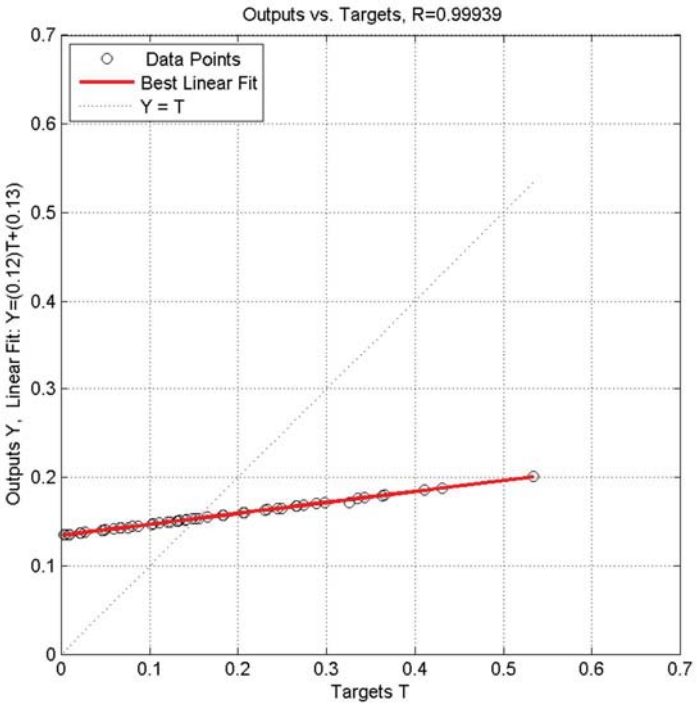

Gráfico 5-1 Dispersión de los datos en peor resultado de simulación.

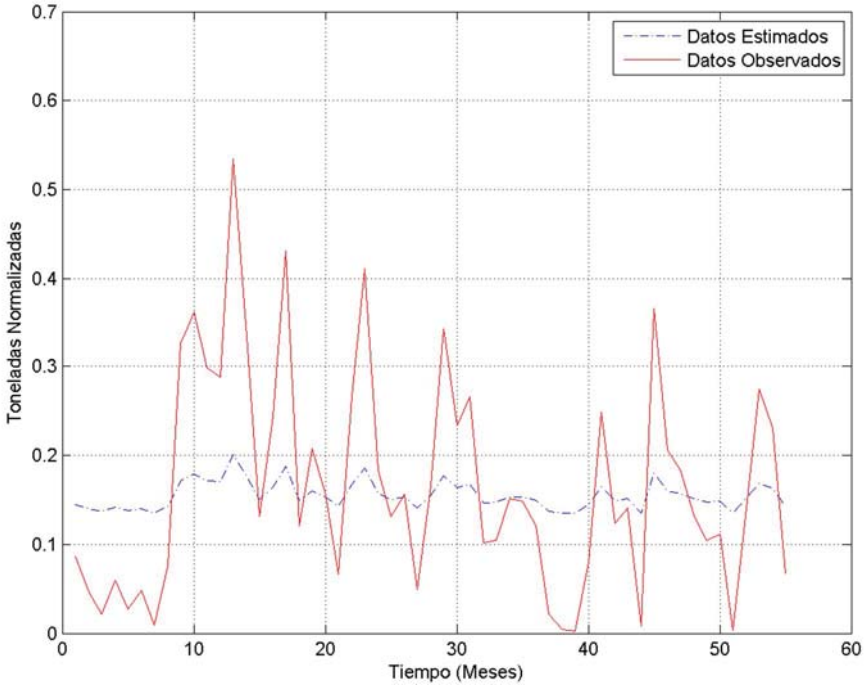

Gráfico 5-2 Simulación del modelo con peores resultados obtenidos por AG.

5.1.2 Resultado promedio en simulación:

Población = 20, iteraciones = 100, MSE= 4.15E-03, R^2=0.99, MAPE=2.023, C= 2.2700, Sig^2=0.0709

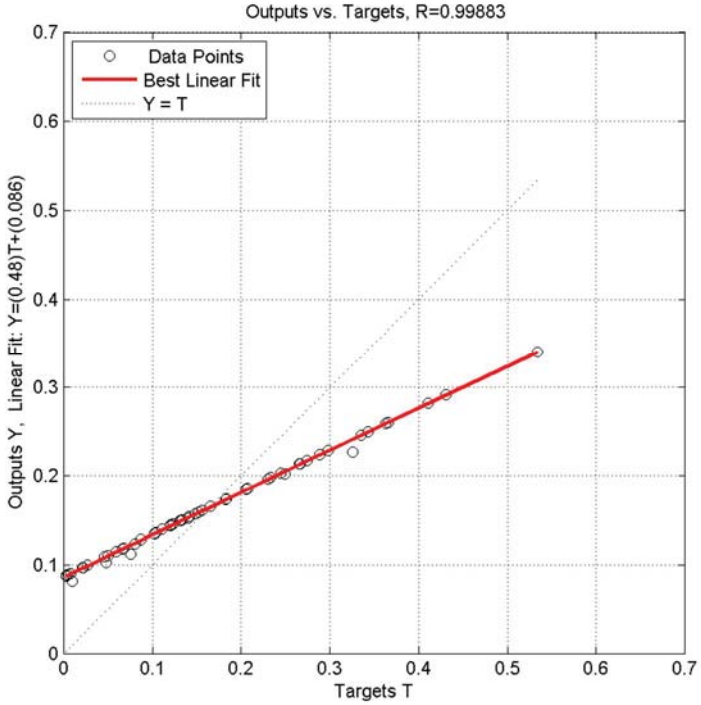

Gráfico 5-3 Dispersión de los datos en resultado promedio de simulación.

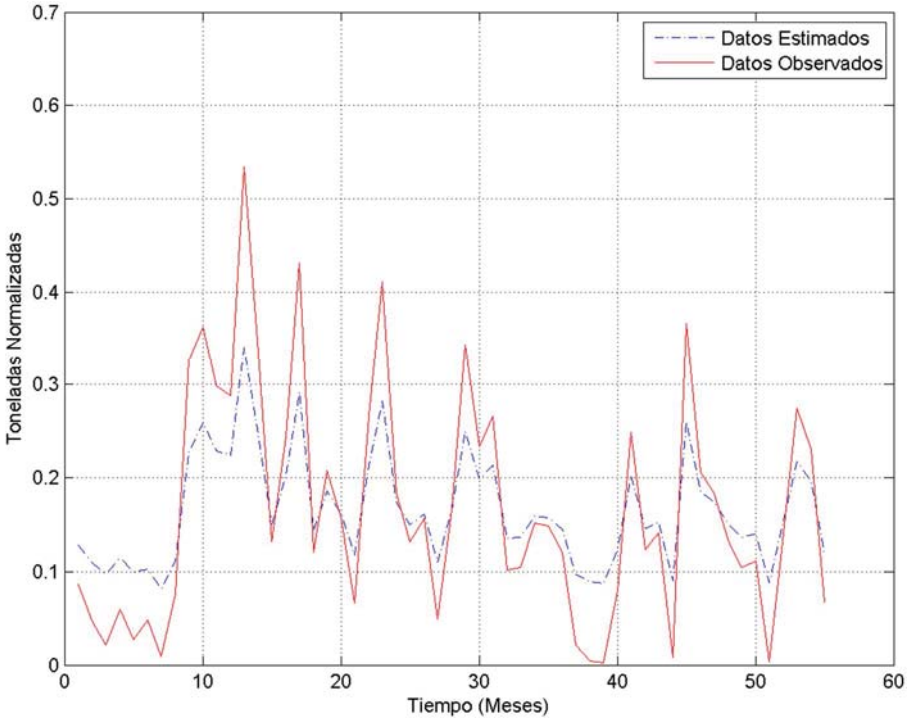

Gráfico 5-4 Simulación del modelo con resultado promedio obtenidos por AG.

5.1.3 Mejor resultado en simulación:

Población = 10, iteraciones = 300, MSE= 1.77E-07, R^2=1, MAPE=0.012, C=730.9824, Sig^2= 0.0316

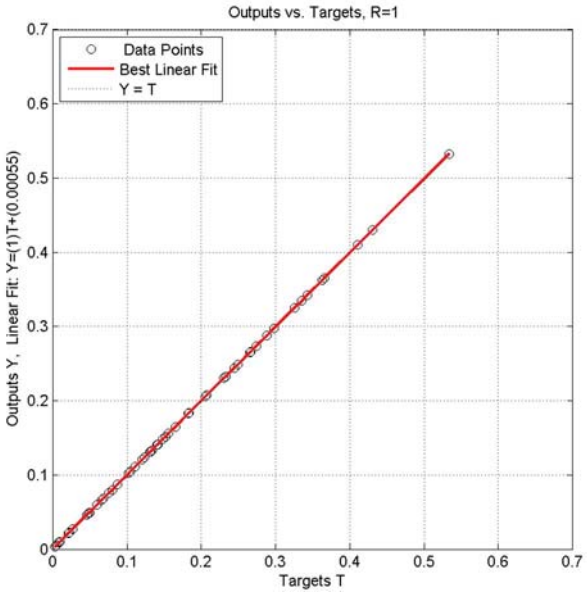

Gráfico 5-5 Dispersión de los datos en mejor resultado de simulación.

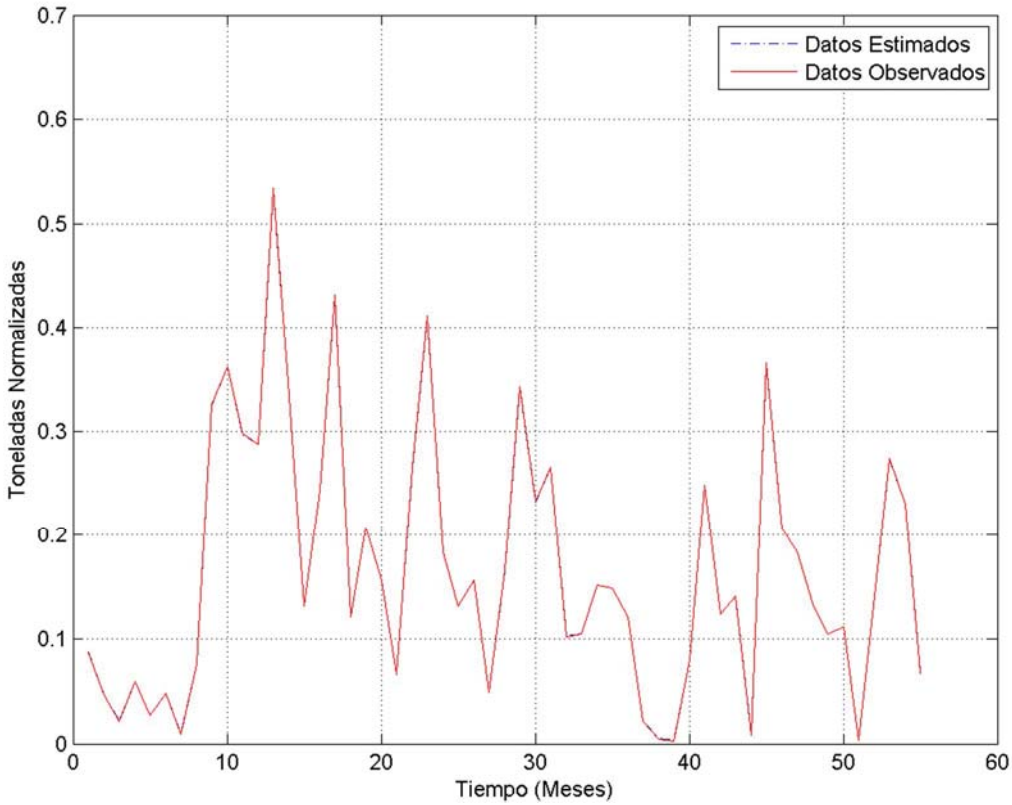

Gráfico 5-6 Simulación del modelo con mejores resultados obtenidos por AG.

5.2 Interpretación de los resultados

Hecho las pruebas de modelo propuesto, se puede visualizar que los resultados obtenidos satisfacen en todos los términos lo esperado. Si bien es cierto, en ocasiones el modelo entrega resultados de baja expectación, nivelando los parámetros que integran al modelo AGSVM, tales como el tamaño de población, datos de entrenamiento, y los parámetros del algoritmo genético, el modelo es capaz de lograr un buen pronóstico, llegando a un $R^2 = 1$, lo que quiere decir que su error de precisión es realmente insignificante.

Como se puede analizar en la tabla 5-1, los distintos resultados obtenidos, son fiel reflejo de los estudios realizados por otros exponentes, quienes declaran que el gran problema de utilizar maquinas vectoriales es determinar sus parámetros de entrada σ y C, y esto lo hemos resuelto utilizando el motor de aprendizaje de los algoritmos genéticos. Esta parte del proceso es la que se encarga de proveer de valores asertivos para σ y C.
El grafico 5-3 muestra el peor resultado de pronóstico obtenido, sin embargo, se puede observar que la figura de los datos pronosticados, mantiene la forma de los datos observados, de la misma manera que en un resultado promedio.
Por otro lado, en el gráfico 5-9, se puede observar que los datos pronosticados son casi 100% iguales a los datos esperados.
La diferencia entre cada prueba se debe a la dispersión de la población inicial, datos que son generados de manera aleatoria.
, obtenida mantiene 11-1, que los resultados de pronóstico están a un nivel que no satisface para nada lo esperado. Esto se debe a que se en cuenta en un cuello de botella, que se produce por los ajuste de los parámetros del algoritmo genético. Una vez que los parámetros sean bien definidos y se establezca una buena categorización de la información, los resultados de éste, se esperan que mantengan un margen de error mínimo.

Visualizando el gráfico 11-2, se puede apreciar, que el margen de error aún es muy elevado para poder pronosticar valores satisfactorios. Se puede concluir que este valor debe ser disminuido lo más cercano a cero para lograr que la generación de cromosomas sean prósperos valores para correctos pronósticos en la máquina de vectores.

Capítulo 6. Conclusión

Hasta los creacionistas encuentran imposible negar que la combinación de la mutación y la selección natural puedan producir adaptación.

Los algoritmos genéticos hacen que esta idea sea insostenible, al demostrar la naturaleza sin junturas del proceso evolutivo. Consideremos, por ejemplo, un problema que consista en programar un circuito para que discrimine entre un tono de 1 kilohercio y un tono de 10 kilohercios, y responda respectivamente con salidas uniformes de 0 y 5 voltios. Digamos que tenemos una solución candidata que puede discriminar con precisión entre los dos tonos, pero sus salidas no son lo bastante uniformes como se requiere; producen pequeñas ondulaciones en lugar del voltaje constante requerido. Supuestamente, de acuerdo con las ideas creacionistas, cambiar este circuito de su estado presente a la solución perfecta sería "microevolución", un cambio pequeño dentro de las capacidades de la mutación y la selección. Pero, sin duda, argumentaría un creacionista, llegar a este mismo estado final desde una ordenación inicial completamente aleatoria de componentes sería "macroevolución", y estaría más allá del alcance de un proceso evolutivo. Sin embargo, los algoritmos genéticos han sido capaces de conseguir ambas cosas: evolucionar el sistema a partir de una ordenación aleatoria hasta la solución casi perfecta y finalmente hasta la solución perfecta y óptima. No surgió ninguna dificultad o brecha insalvable en ningún punto del camino. En ningún momento hizo falta intervención humana para montar un conjunto de componentes irreduciblemente complejo (a pesar del hecho de que el producto final sí contiene tal cosa) o para "guiar" al sistema evolutivo a través de algún pico dificultoso. El circuito evolucionó, sin la ayuda de ninguna orientación inteligente, desde un estado completamente aleatorio y no funcional hasta un estado rigurosamente complejo, eficiente y óptimo.

Quedó evidenciado que existen métodos que ayudarían a la selección y mutación de los genes para obtener candidatos superiores, que el tiempo que toma en calibrar los datos se optimiza a través de los algoritmos genéticos.

Se evidencia que los resultados obtenidos a entrenados en SVM muestran los resultados esperados, con un margen de error ínfimo.

Capítulo 7. Bibliografía

1 "Adaptive Learning: Fly the Brainy Skies." *Wired*, vol.10, no.3 (marzo de 2002).

2 Altshuler, Edward y Derek Linden. "Design of a wire antenna using a genetic algorithm." *Journal of Electronic Defense*, vol.20, no.7, p.50-52 (julio de 1997).

3 Andre, David y Astro Teller. "Evolving team Darwin United." En *RoboCup-98: Robot Soccer World Cup II*, Minoru Asada and Hiroaki Kitano (eds). Lecture Notes in Computer Science, vol.1604, p.346-352. Springer-Verlag, 1999.

Ver también: Willihnganz, Alexis. "Software that writes software." *Salon*, 10 de agosto de 1998.

4 Andreou, Andreas, Efstratios Georgopoulos y Spiridon Likothanassis. "Exchange-rates forecasting: A hybrid algorithm based on genetically optimized adaptive neural networks." *Computational Economics*, vol.20, no.3, p.191-210 (diciembre de 2002).

5 Ashley, Steven. "Engineous explores the design space." *Mechanical Engineering*, febrero de 1992, p.49-52.

6 Assion, A., T. Baumert, M. Bergt, T. Brixner, B. Kiefer, V. Seyfried, M. Strehle y G. Gerber. "Control of chemical reactions by feedback-optimized phase-shaped femtosecond laser pulses." *Science*, vol.282, p.919-922 (30 de octubre de 1998).

7 Au, Wai-Ho, Keith Chan, y Xin Yao. "A novel evolutionary data mining algorithm with applications to churn prediction." *IEEE Transactions on Evolutionary Computation*, vol.7, no.6, p.532-545 (diciembre de 2003).

8 Beasley, J.E., J. Sonander y P. Havelock. "Scheduling aircraft landings at London Heathrow using a population heuristic." *Journal of the Operational Research Society*, vol.52, no.5, p.483-493 (mayo de 2001).

9 Begley, Sharon y Gregory Beals. "Software au naturel." *Newsweek*, 8 de mayo de 1995, p.70.

10 Benini, Ernesto y Andrea Toffolo. "Optimal design of horizontal-axis wind turbines using blade-element theory and evolutionary computation." *Journal of Solar Energy Engineering*, vol.124, no.4, p.357-363 (noviembre de 2002).

11 Burke, E.K. y J.P. Newall. "A multistage evolutionary algorithm for the timetable problem." *IEEE Transactions on Evolutionary Computation*, vol.3, no.1, p.63-74 (abril de 1999).

12 Charbonneau, Paul. "Genetic algorithms in astronomy and astrophysics." *The Astrophysical Journal Supplement Series*, vol.101, p.309-334 (diciembre de 1995).

13 Chellapilla, Kumar y David Fogel. “Evolving an expert checkers playing program without using human expertise.” *IEEE Transactions on Evolutionary Computation*, vol.5, no.4, p.422-428 (agosto de 2001).

14 Chellapilla, Kumar y David Fogel. “Anaconda defeats Hoyle 6-0: a case study competing an evolved checkers program against commercially available software.” En *Proceedings of the 2000 Congress on Evolutionary Computation*, p.857-863. IEEE Press, 2000.

15 Chellapilla, Kumar y David Fogel. “Verifying Anaconda's expert rating by competing against Chinook: experiments in co-evolving a neural checkers player.” *Neurocomputing*, vol.42, no.1-4, p.69-86 (enero de 2002).

16 Chryssolouris, George y Velusamy Subramaniam. “Dynamic scheduling of manufacturing job shops using genetic algorithms.” *Journal of Intelligent Manufacturing*, vol.12, no.3, p.281-293 (junio de 2001).

17 Coale, Kristi. “Darwin in a box.” *Wired News*, 14 de julio de 1997.

18 Coello, Carlos. “An updated survey of GA-based multiobjective optimization techniques.” *ACM Computing Surveys*, vol.32, no.2, p.109-143 (junio de 2000).

19 Davidson, Clive. “Creatures from primordial silicon.” *New Scientist*, vol.156, no.2.108, p.30-35 (15 de noviembre de 1997).

20 Dawkins, Richard. *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design*. W.W. Norton, 1996.

21 Dembski, William. *No Free Lunch: Why Specified Complexity Cannot Be Purchased Without Intelligence*. Rowman & Littlefield, 2002.

22 Fleming, Peter y R.C. Purshouse. “Evolutionary algorithms in control systems engineering: a survey.” *Control Engineering Practice*, vol.10, p.1.223-1.241 (2002).

23 Fonseca, Carlos y Peter Fleming. “An overview of evolutionary algorithms in multiobjective optimization.” *Evolutionary Computation*, vol.3, no.1, p.1-16 (1995).

24 Forrest, Stephanie. “Genetic algorithms: principles of natural selection applied to computation.” *Science*, vol.261, p.872-878 (1993).

25 Gibbs, W. Wayt. “Programming with primordial ooze.” *Scientific American*, octubre de 1996, p.48-50.

26 Gillet, Valerie. “Reactant- and product-based approaches to the design of combinatorial libraries.” *Journal of Computer-Aided Molecular Design*, vol.16, p.371-380 (2002).

27 Giro, R., M. Cyrillo y D.S. Galvão. “Designing conducting polymers using genetic algorithms.” *Chemical Physics Letters*, vol.366, no.1-2, p.170-175 (25 de noviembre de 2002).

28 Glen, R.C. y A.W.R. Payne. “A genetic algorithm for the automated generation of molecules within constraints.” *Journal of Computer-Aided Molecular Design*, vol.9, p.181-202 (1995).

29 Goldberg, David. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

30 Graham-Rowe, Duncan. “Radio emerges from the electronic soup.” *New Scientist*, vol.175, no.2.358, p.19 (31 de agosto de 2002).

Ver también: Bird, Jon y Paul Layzell. “The evolved radio and its implications for modelling the evolution of novel sensors.” En *Proceedings of the 2002 Congress on Evolutionary Computation*, p.1.836-1.841.

31 Graham-Rowe, Duncan. “Electronic circuit 'evolves' from liquid crystals.” *New Scientist*, vol.181, no.2.440, p.21 (27 de marzo de 2004).

32 Haas, O.C.L., K.J. Burnham y J.A. Mills. “On improving physical selectivity in the treatment of cancer: A systems modelling and optimisation approach.” *Control Engineering Practice*, vol.5, no.12, p.1.739-1.745 (diciembre de 1997).

33 Hanne, Thomas. “Global multiobjective optimization using evolutionary algorithms.” *Journal of Heuristics*, vol.6, no.3, p.347-360 (agosto de 2000).

34 Haupt, Randy y Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, 1998.

35 He, L. y N. Mort. “Hybrid genetic algorithms for telecommunications network back-up routeing.” *BT Technology Journa*l, vol.18, no.4, p. 42-50 (octubre de 2000).

36 Holland, John. “Genetic algorithms.” *Scientific American*, julio de 1992, p. 66-72.

37 Hughes, Evan y Maurice Leyland. “Using multiple genetic algorithms to generate radar point-scatterer models.” *IEEE Transactions on Evolutionary Computation*, vol.4, no.2, p.147-163 (julio de 2000).

38 Jensen, Mikkel. “Generating robust and flexible job shop schedules using genetic algorithms.” *IEEE Transactions on Evolutionary Computation*, vol.7, no.3, p.275-288 (junio de 2003).

39 Kewley, Robert y Mark Embrechts. “Computational military tactical planning system.” *IEEE Transactions on Systems, Man and Cybernetics, Part C - Applications and Reviews*, vol.32, no.2, p.161-171 (mayo de 2002).

40 Kirkpatrick, S., C.D. Gelatt y M.P. Vecchi. “Optimization by simulated annealing.” *Science*, vol.220, p.671-678 (1983).

41 Koza, John, Forest Bennett, David Andre y Martin Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers, 1999.

42 Koza, John, Martin Keane, Matthew Streeter, William Mydlowec, Jessen Yu y Guido Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

Ver también: Koza, John, Martin Keane y Matthew Streeter. "Evolving inventions." *Scientific American*, febrero de 2003, p. 52-59.

43 Keane, A.J. y S.M. Brown. "The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques." En *Adaptive Computing in Engineering Design and Control '96 - Proceedings of the Second International Conference*, I.C. Parmee (ed), p.107-113. University of Plymouth, 1996.

Ver también: Petit, Charles. "Touched by nature: Putting evolution to work on the assembly line." *U.S. News and World Report*, vol.125, no.4, p.43-45 (27 de julio de 1998

44 Lee, Yonggon y Stanislaw H. Zak. "Designing a genetic neural fuzzy antilock-brake-system controller." *IEEE Transactions on Evolutionary Computation*, vol.6, no.2, p.198-211 (abril de 2002).

45 Lemley, Brad. "Machines that think." *Discover*, enero de 2001, p.75-79.

46 Mahfoud, Sam y Ganesh Mani. "Financial forecasting using genetic algorithms." *Applied Artificial Intelligence*, vol.10, no.6, p.543-565 (1996).

47 Mitchell, Melanie. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

48 Naik, Gautam. "Back to Darwin: In sunlight and cells, science seeks answers to high-tech puzzles." *The Wall Street Journal*, 16 de enero de 1996, p. A1.

49 Obayashi, Shigeru, Daisuke Sasaki, Yukihiro Takeguchi, y Naoki Hirose. "Multiobjective evolutionary computation for supersonic wing-shape optimization." *IEEE Transactions on Evolutionary Computation*, vol.4, no.2, p.182-187 (julio de 2000).

50 Petzinger, Thomas. "At Deere they know a mad scientist may be a firm's biggest asset." *The Wall Street Journal*, 14 de julio de 1995, p.B1.

Ver también: "Evolving business, with a Santa Fe Institute twist." *SFI Bulletin*, invierno de 1998.

51 Porto, Vincent, David Fogel y Lawrence Fogel. "Alternative neural network training methods." *IEEE Expert*, vol.10, no.3, p.16-22 (junio de 1995).

52 Rao, Srikumar. "Evolution at warp speed." *Forbes*, vol.161, no.1, p.82-83 (12 de enero de 1998).

53 Rizki, Mateen, Michael Zmuda y Louis Tamburino. "Evolving pattern recognition systems." *IEEE Transactions on Evolutionary Computation*, vol.6, no.6, p.594-609 (diciembre de 2002).

54 Robin, Franck, Andrea Orzati, Esteban Moreno, Otte Homan, y Werner Bachtold. "Simulation and evolutionary optimization of electron-beam lithography with genetic and simplex-downhill

algorithms." *IEEE Transactions on Evolutionary Computation*, vol.7, no.1, p.69-82 (febrero de 2003).

55 Sagan, Carl. *Broca's Brain: Reflections on the Romance of Science*. Ballantine, 1979.

56 Sambridge, Malcolm y Kerry Gallagher. "Earthquake hypocenter location using genetic algorithms." *Bulletin of the Seismological Society of America*, vol.83, no.5, p.1.467-1.491 (octubre de 1993).

57 Sasaki, Daisuke, Masashi Morikawa, Shigeru Obayashi y Kazuhiro Nakahashi. "Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms." En *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001, Zurich, Switzerland, March 2001: Proceedings*, K. Deb, L. Theile, C. Coello, D. Corne y E. Zitler (eds). Notas de la confenrencia en *Computer Science*, vol.1993, p.639-652. Springer-Verlag, 2001.

58 Sato, S., K. Otori, A. Takizawa, H. Sakai, Y. Ando y H. Kawamura. "Applying genetic algorithms to the optimum design of a concert hall." *Journal of Sound and Vibration*, vol.258, no.3, p. 517-526 (2002).

59 Schechter, Bruce. "Putting a Darwinian spin on the diesel engine." *The New York Times*, 19 de septiembre de 2000, p. F3.

Ver también: Patch, Kimberly. "Algorithm evolves more efficient engine." *Technology Research News*, junio/julio de 2000.

60 Srinivas, N. y Kalyanmoy Deb. "Multiobjective optimization using nondominated sorting in genetic algorithms." *Evolutionary Computation*, vol.2, no.3, p.221-248 (otoño de 1994).

61 Soule, Terrence y Amy Ball. "A genetic algorithm with multiple reading frames." En *GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference*, Lee Spector y Eric Goodman (eds). Morgan Kaufmann, 2001.

62 Tang, K.S., K.F. Man, S. Kwong y Q. He. "Genetic algorithms and their applications." *IEEE Signal Processing Magazine*, vol.13, no.6, p.22-37 (noviembre de 1996).

63 Weismann, Dirk, Ulrich Hammel, y Thomas Bäck. "Robust design of multilayer optical coatings by means of evolutionary algorithms." *IEEE Transactions on Evolutionary Computation*, vol.2, no.4, p.162-167 (noviembre de 1998).

64 Williams, Edwin, William Crossley y Thomas Lang. "Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm." *Journal of the Astronautical Sciences*, vol.49, no.3, p.385-400 (julio-septiembre de 2001).

Ver también: "Selecting better orbits for satellite constellations." Spaceflight Now, 18 de octubre de 2001.

"Darwinian selection of satellite orbits for military use." Space.com, 16 de octubre de 2001.

65 Zitzler, Eckart y Lothar Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the Strength Pareto approach." *IEEE Transactions on Evolutionary Computation*, vol.3, no.4, p.257-271 (noviembre de 1999).

66 J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimum Optimization," in Schölkopf, Burges and Smola, Eds., Advances in Kernel Methods— Support Vector Learning, Cambridge MA: MIT Press, 1998, pp 185-208.

67 Martin, M. (2002). On-line support vector machines for function approximation. (Tech. Rep. LSI-02-11-R).Catalunya, Spain: Software Department, Universitat Politecnica de Catalunya.

68 Las maquinas de vectores de soporte para identificación en línea (Juan Angel Resendiz Trejo, sep 2006, México)

69 Smola AJ. Learning with kernels. PhD thesis, Department of Computer Science. Germany: Technical University Berlin; 1998.

[**70**] Deb K, Agrawal RB. Simulated binary crossover for continuous search space. Complex Sys 1995;9(2):115–48.

[**71**] Deb K, Kumar A. Real-coded genetic algorithms with simulated binary crossover: studies on multimodal and multiobjective problems. Complex Sys 1995;9(6):431–54.