PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

PRONÓSTICO DE CAPTURA DE ANCHOVETAS DE LA ZONA NORTE DE CHILE USANDO VECTOR DE SOPORTE AUTOREGRESIVO

CARLOS ANDRÉS COSMING GONZÁLEZ

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

ABRIL 2012

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

PRONÓSTICO DE CAPTURA DE ANCHOVETAS DE LA ZONA NORTE DE CHILE USANDO VECTOR DE SOPORTE AUTOREGRESIVO

CARLOS ANDRÉS COSMING GONZÁLEZ

Profesor Guía: **Nibaldo Rodriguez Agurto**

Carrera: **Ingeniería Civil Informática**

ABRIL 2012

Dedicatoria

Dedicada a mi madre, a mi padre y a mi hermana, por su gran esfuerzo, confianza y apoyo incondicional.

Agradecimientos

A mi familia, compañeros y amigos, a mi profesor guía y a la Universidad.

Índice General

Índice de Figuras

Índice de Tablas

Índice de algoritmos

Abreviaciones y Nomenclatura

ANN	Redes Neuronales Artificiales — del inglés Artificial Neural Networks —
pbest	Mejor valor de *fitness* encontrado por la partícula hasta el momento del algoritmo PSO.
$\mathscr{D}$	Conjunto de Entrenamiento — del inglés DataSet —
DAPSO	Particle Swarm Optimizer with Dynamic Adaptation
ξ	Variable de Holgura para Modelos no Lineales de Vectores Soporte
ERM	Minimización del Riesgo Empírico — del inglés Empirical Risk Minimization —
$\mathscr{F}$	Espacio de Características — del inglés, Feature Space —
gbest	Mejor valor de *fitness* encontrado por alguna partícula del enjambre del algoritmo PSO.
GRBF	Función Gaussiana de Base Radial — del inglés Gaussian Radial Basis Function —
$\mathscr{H}$	Hiperplano Separador entre la clase $\mathscr{C}_-$ y la clase $\mathscr{C}_+$
Π	Hiperplano Separador Óptimo entre la clase $\mathscr{C}_-$ y la clase $\mathscr{C}_+$
K	Función Kernel
KKT	Condiciones de Karush-Kuhn-Tucker : son condiciones necesarias y suficientes para que la solución de una programación no lineal séa óptima.
$K(x_i, x_j)$	Función Kernel : Calcula directamente el producto interno entre dos vectores del espacio característico en un espacio de mayor dimensionalidad, sin necesidad de realizar el mapeo a dicho espacio.
ℓ	Número de elementos que constituyen el Conjunto de Entrenamiento $\mathscr{T}$
lbest	Mejor valor de *fitness* encontrado en el vecindario de una partícula del algoritmo PSO.

L_ε	Función de Pérdida $\varepsilon - Insensitive$
$\mathscr{LM}$	Espacio de Aproximación Generado por la Arquitectura de las LMs
LS-SVM + PSO	Modelo de Máquina de Soporte Vectorial con Optimización de Parámetros mediante Algoritmo por Enjambre de Partículas.
MLP	Perceptrón Multicapa — del inglés Multilayer Perceptron —
$\Phi(\cdot)$	Función Transformadora del Espacio de Entrada a un nuevo Espacio de mayor Dimensionalidad — Espacio Característico $\mathscr{F}$ —
OSH	Hiperplano Óptimo de Separación — del inglés Optimum Separation Hyperplane —
PGAE	Problema General de Aprendizaje a partir de Ejemlos: Problema de elegir entre el conjunto establecido de funciones $f(x,\omega) \in \mathscr{LM}$ aquella que minimice el funcional de riesgo.
PSO-IW	Particle Swarm Optimization with Inertia Weight
PSO	Optimización por Enjambre de Partículas — del inglés Particle Swarm Optimization —
QP	Programación Cuadrática — del inglés Quadratic Programming —
QPSO	Quantum Particle Swarm Optimization
RBF	Función de Base Radial — del inglés Radial Basis Function —
$R_{emp}(\omega)$	Funcional de Riesgo Empírico
R_{emp}^{reg}	Funcional de Riesgo Empírico Ordinal
$R^{reg}(\omega)$	Funcional de Riesgo Regularizado
$R_{srm}(\omega)$	Funcional de Riesgo Estructural
$R(\omega)$	Funcional de Riesgo Teórico: Medida de discrepancia o función de coste entre la resupuesta del sistema, y, a una entrada, x, y la respuesta producida por la máquina de aprendizaje
SI	Inteligencia Colectiva — del inglés Swarm Intelligence —
SLT	Teoría del Aprendizaje Estadístico — del inglés Statiscal Learning Theory —
SRM	Minimización del Riesgo Estructural — del inglés Structural Risk Minimization —
SVMC	Máquina de Soporte Vectorial para Clasificación —del inglés Support Vector Machine for Classification —

SVMR	Máquina de Soporte Vectorial para Regresión — del inglés Support Vector Machine for Regression —
SVM	Máquinas de Soporte Vectorial — del inglés Support Vector Machines —
SV	Vectores Soporte — del inglés Support Vectors —
$\mathscr{T}$	Conjunto de Entranamiento, contenie los vectores de entrada y la respuesta del sistema.
$\overrightarrow{v}$	Representa la velocidad de la partícula del algoritmo PSO
VC	Dimensión Vapnik Chervonenkis — del inglés Vapnik Chervonenkis Dimension —, es una medida de la capacidad de los algoritmos de clasificación estadística, definida como la cardinalidad del mayor conjunto de puntos que el algoritmo puede separar.
$\overrightarrow{x}$	Vector de ubicación de una partícula del algoritmo PSO. También representa el valor de la función *fitness* que mide la calidad de la solución.
X	Espacio de Entrada
Y	Espacio de Salida

Resumen

Entre las actividades extractivas nacionales, la pesca representa uno de los sectores más significativos, por su aporte a la economía nacional y por el progresivo crecimiento tanto en la extracción misma, como en la industria derivada.

Uno de los principales factores de incertidumbre que afecta al sector pesquero industrial es la disponibilidad de recursos, donde la escasez de biomasa es el principal factor que puede entorpecer y restringir el desarrollo sustentable del sector.

Este proyecto define un modelo de pronóstico de captura de anchovetas de la zona norte de Chile a través de una Máquina de Soporte Vectorial con Algoritmos por Enjambre de Partículas para la optimización de parámetros del modelo. El modelo propuesto LS-SVM + LAPSO, presenta un 94,63 % de la varianza explicada del problema, con esto se espera entregar una nueva herramienta para el control y gestión de los recursos de este importante sector industrial nacional.

Palabras Clave: Engraulis Ringens, Anchoveta, Regresión, Modelo de Pronóstico, Máquinas de Soporte Vectorial, Aprendizaje Supervisado, Kernels, Optimización por Enjambre de Partículas, Optimización.

Abstract

Among the national extraction activities, fishing is one of the most significant sectors for their contribution to the national economy and the progressive increase in the extraction itself, as in the related industries.

One of the main uncertainties affecting the commercial fishing industry is the availability of resources, where the shortage of biomass is the main factor that may hinder and restrict the sustainable development of the sector.

This project defines a forecast model for the anchovy catch in northern of Chile, through a Support Vector Machine with Particle Swarm Algorithms for optimization of model parameters. The proposed model LS-SVM + LAPSO presents a 94.63 % of the explained variance of the problem, with this is expected to deliver a new tool for controlling and managing the resources of this important national industry.

Keywords: Engraulis Ringens, Anchovy, Regression, Forecast Model, Support Vector Machines, Supervised Learning, Kernels, Particle Swarm Optimization, Optimization.

Capítulo 1

Introducción

El sector pesca es uno de los más importantes dentro de la economía nacional. A pesar de mantener una baja participación sobre el Producto Interno Bruto y nivel de empleo, es la tercera actividad en generación de divisas, luego del sector minero y forestal, y la segunda sobre la base de recursos renovables. Durante los últimos cinco años, el sector pesca ha alcanzado una participación promedio anual de 6,8 % sobre las exportaciones totales del país.

No cabe duda que los últimos tiempos han sido bastantes turbulentos para el sector pesquero y acuícola. Cuando los problemas sanitarios que venían afectando el cultivo de salmón atlántico comenzaban a mostrar un cambio de tendencia, el sector pesquero se vio seriamente afectado por el terremoto y posterior tsunami que azotó a la zona centro–sur del país el 27 de febrero de 2010, generando severos daños sobre embarcaciones y plantas productivas, reflejándose en los niveles de desembarques, producción y exportación, estimándose fuertes caídas en 2010 y 2011 [Donoso, 2010].

Dentro del sector extractivo, destacan por su importancia los recursos pelágicos, entre ellos se encuentran la **anchoveta** — Engraulis Ringens — que aporta el 22,2 % del total de desembarques acumulados a la fecha [Sectorial, 2010]. La zona norte del pais — regiones III y IV — concentraron el 5,4 % de los desembarques [Donoso, 2010].

Uno de los principales factores de incertidumbre que afecta al sector pesquero industrial es la disponibilidad de recursos pesqueros, donde la escasez de biomasa — **stock** — es el principal factor que puede entorpecer y restringir el desarrollo sustentable del sector, y por ende el crecimiento de la actividad de las empresas que participan en la industria, a través de la asignación de menores **cuotas** de captura.

De esta manera, se hace necesario realizar una constante evaluación de los recursos pesqueros, que consiste en estimar la biomasa de un recurso en el tiempo y su correspondiente estructura, es decir su composición de edades o tallas. Esta evaluación es una tarea compleja e incierta, considerando que estos recursos no se encuentran visibles directamente, se desplazan continuamente y se desarrollan en un medio extremo dinámico,

es decir, en diferentes masas de agua y diferentes niveles de productividad. A pesar de dichas complejidades, se dispone de métodos y procedimientos para lograr dicho fin. La debilidad de estos métodos, por la cantidad de información que requieren y por la complejidad de los procesos que es necesario modelar, es que establecen el diagnóstico de un recurso con un desfase de tiempo, pudiendo por lo tanto determinarse el estado de un determinado recurso en base a un nivel de biomasa que ha presentado una variación posterior a su evaluación, considerando la dinámica propia que presentan las pesquerías.

El ecosistema que rodea a la anchoveta se caracteriza por su complejidad y comportamiento no lineal. La variabilidad de su nivel de abundancia se ve afectada directamente por una serie de variables ambientales, entre las cuales se encuentran: La corriente del Niño, temperatura superficial del mar y la concentración de alimento (plancton).

Enfoques tradicionales para el estudio de datos empíricos han sufrido dificultades con la generalización, teniendo así una gran susceptibilidad a producir modelos altamente sobreajustados a los datos. Esto es una consecuencia de los algoritmos de optimización utilizados para la selección de los parámetros y las medidas estadísticas para seleccionar el mejor modelo.

Las máquinas de soporte vectorial — SVM, del inglés Support Vector Machine —, están ganando popularidad debido a un conjunto atractivo de características y comportamiento empírico muy prometedor. La formulación encarna el principio de minimización del riesgo estructural, que ha demostrado ser superior al ya tradicional principio de minimización de riesgo empírico, empleado por las redes neuronales convencionales [S.R. Gunn and Bossley, 1997]. Es esta diferencia la que equipa a las SVMs con mayor capacidad de generalizar, que es el objetivo del aprendizaje estadístico. Las SVMs se desarrollaron para resolver el problema de clasificación, pero recientemente se han ampliado al ámbito de los problemas de regresión [Cortes and Vapnik, 1995] y [Smola and Schölkopf, 1998], que será el enfoque con el cual este proyecto será fundamentado.

En función de lo anteriormente expuesto, es que se propondrá un modelo de pronóstico basado en máquinas de soporte vectorial autoregresivas para la captura de anchovetas de la zona norte Chile, con el afán de presentar una nueva alternativa a las herramientas ya existentes para el manejo del stock de la biomasa y posterior asignación de cuotas de captura para esta zona geográfica del país.

1.1 Objetivos

A continuación se darán a conocer los objetivos planteados para este proyecto. Por un lado se hará mención al objetivo general del proyecto, seguido de objetivos específicos que servirán de apoyo a la obtención del objetivo general.

1.1.1 Objetivo General

Desarrollar un modelo de regresión basado en una Máquina de Soporte Vectorial para predecir la captura de anchovetas de la zona norte de Chile, utilizando algoritmos de Optimización por Enjambre de Partículas.

1.1.2 Objetivos Específicos

- Explicar el marco teórico de las Máquinas de Soporte Vectorial y Optimización por Enjambre de Partículas.
- Diseñar la estructura y estimar los parámetros del modelo SVM mediante el uso de PSO y sus variantes.
- Implementar y evaluar la precisión de los modelos predicción propuestos.

1.2 Estructura del Documento

El Capítulo 2 describe todos los conceptos relacionados a la teoría del *Problema de Aprendizaje a Partir de Ejemplos*, definiendo el problema general y sus casos particulares: estimación de regresión, clasificación o reconocimiento de patrones y regresión ordinal y cómo éstas a través de un algoritmo de aprendizaje pueden plantear un modelo que permita solucionar el problema. El Capítulo 3 hace referencia a los fundamentos de las *Máquinas de Soporte Vectorial*, describiendo sus principales componentes y derivaciones, ya sea para los problemas de clasificación como para los de regresión, además se presentará la técnica de mínimos cuadrados para las SVM, que permiten solucionar el problema de mejor manera gracias a su nuevo enfoque de encontrar las soluciones, lo que generará beneficios en su implantación. El Capítulo 4 aborda todos los conceptos de la *Optimización por Enjambre de Partículas*, describiendo su funcionamiento y variantes a lo que su algoritmo respecta, que básicamente se relacionan a la variación e integración de parámetros. El Capítulo 5 define el modelo general propuesto, denominado LS-SVM + PSO, presentando su estructura, el manejo de los datos y las métricas utilizadas para su posterior evaluación. El Capítulo 6 expone los resultados obtenidos de los modelos propuestos. Por último se encuentra el Capítulo 7, que corresponde a las conclusiones obtenidas a partir de este proyecto de título, destacando los aspectos más importantes de los capítulos anteriores y cómo las técnicas abordadas permitieron plantear un modelo de regresión para predecir las capturas de anchoveta de la zona norte de Chile.

Capítulo 2

Problema de Aprendizaje a partir de Ejemplos

2.1 Introducción

El objetivo fundamental del estudio de datos empíricos, es aprender a partir de estos datos y para ello se busca la existencia de alguna dependencia funcional entre un conjunto de vectores inputs — o de entrada — .

$$\{x_i, i = 1, \ldots, n\} \subseteq X \subseteq \mathbb{R}^d. \tag{2.1.1}$$

y valores outputs — o de salida —.

$$\{y_i, i = 1, \ldots, n\} \subseteq Y \subseteq \mathbb{R}. \tag{2.1.2}$$

Es por esto que se hace necesario y muy pertinente describir el proceso de aprendizaje que permite enunciar el *Problema General de Aprendizaje a partir de Ejemplos* como la minimización funcional de riesgo sobre la base de datos empíricos. Según la naturaleza de los valores en los datos de salida, el problema general da lugar al enunciado de diferentes tareas de aprendizaje, una de ellas corresponde a la estimación de regresiones, el reconocimiento de patrones o la regresión ordinal.

Una vez definido el problema general de aprendizaje supervisado y el caso particular derivado a enfrentar en este trabajo, es que surge la necesidad de establecer un método que permita calcular una adecuada solución. Este método dependerá de una serie de factores que condicionarán su habilidad para generalizar. La elección del principio inductivo sobre el que se sienta la base del método de aprendizaje, resultará crítica para conseguir el resultado deseado.

2.2 Planteamiento del Problema General de Aprendizaje

Tomando como punto de referencia los trabajos [Vapnik, 1995], [Vapnik, 1998] y [Cherkassky and Mulier, 1998], se entenderá por *Aprendizaje a Partir de Ejemplos* — APE en corto — al proceso de estimar una dependencia desconocida entre los datos de entrada X y los de salida Y de un sistema utilizando un número limitado de observaciones. El modelo general de un proceso de aprendizaje a partir de ejemplos se desarrolla sobre tres componentes:

- Un *generador* de vectores de entrada aleatorios, $x \in X \subseteq \mathbb{R}^d$ — en el caso real multivariable —, producidos de forma no controlable por el investigador de forma independiente e idénticamente distribuida — en corto i.i.d. — a partir de una densidad de probabilidad $p(x)$ fijada pero desconocida.
- Un *sistema* que retorna un valor de salida, $y \in Y \subseteq \mathbb{R}$, para todo vector de entrada x siguiendo una densidad condicional $p(y|x)$ también fijada pero desconocida. Este modelo general incluiría por ejemplo el caso de un sistema determinista que utilice alguna función $y = f(x) + \varepsilon$, donde ε es un ruido aleatorio de valor medio nulo, también denominado ruido blanco.
- Una *máquina de aprendizaje* — LM, del inglés *Learning Machine* — capaz de desarrollar un espacio de funciones.

$$\mathscr{LM} = \left\{ f(x, \omega) : x \in X \subseteq \mathbb{R}^d, \omega \in \Omega \right\}, \tag{2.2.1}$$

  donde Ω es un conjunto de parámetros abstracto.

La máquina de aprendizaje observa ℓ pares que constituyen el *Conjunto de Entrenamiento* $\mathscr{T}$ conteniendo los vectores de entrada y la respuesta del sistema

$$\mathscr{T} = \left\{ (x_p, y_p)_{p=1}^{\ell} \subset X \times Y \sim P_{xy}^{\ell} \right\}, \tag{2.2.2}$$

para construir a partir de ellos durante este período, denominado *período de entrenamiento*, algún operador que sirva de predictor de respuestas del sistema a entradas específicas producidas por el generador.

El *Problema General de Aprendizaje* deberá definirse como aquel de elegir entre el conjunto establecido de funciones $f(x, \omega) \in \mathscr{LM}$ aquella que posea menor discrepancia con la repuesta del sistema.

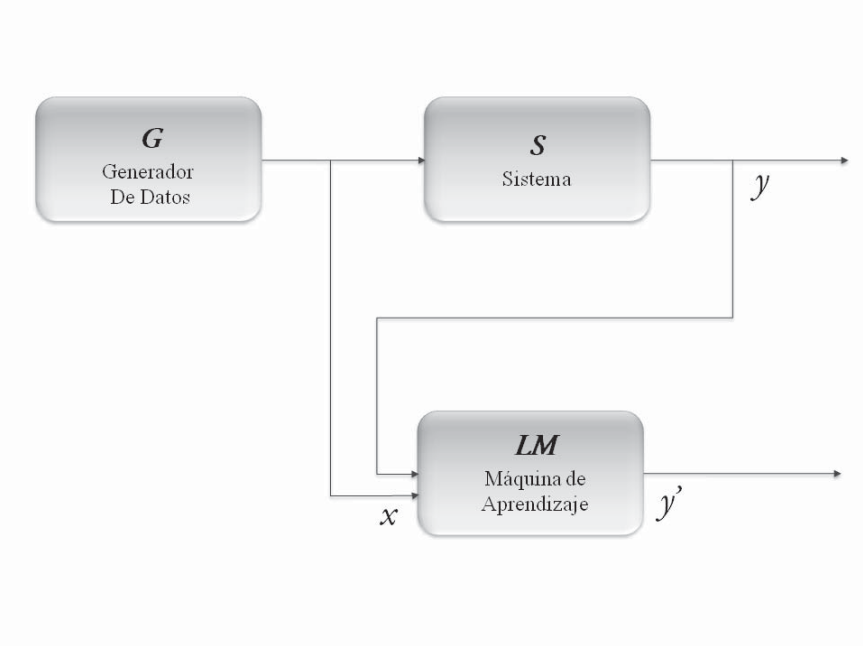


Figura 2.1: *Modelo General de un proceso de aprendizaje a partir de ejemplos.*

Definición 2.2.1. *Sea $L(y, f(x, \omega))$ una medida de discrepancia o función de coste entre la respuesta del sistema, y, a una entrada, x, y la respuesta producida por la máquina de aprendizaje $f(x, \omega)$. Sea el Funcional de Riesgo $R(\omega)$ la esperanza estadística de esta discrepancia.*

$$R(\omega) = \int L(y, f(x, \omega))p(x, y), \tag{2.2.3}$$

entonces se define el Problema General de Aprendizaje a partir de Ejemplos — PGAE en corto — como aquel de elegir entre el conjunto establecido de funciones $f(x, \omega) \in \mathscr{LM}$ aquella que minimice el funcional de riesgo.

$$f(x, \omega^{\mathscr{LM}}) = \underset{\omega \in \Omega}{arg\,min} R(w), \tag{2.2.4}$$

cuando la densidad de probabilidad conjunta $p(x, y)$ es desconocida y la única información accesible está contenida en el conjunto de entrenamiento $\mathscr{T}$.

Observando la definición, puede establecerse que la función elegida durante el proceso de aprendizaje debe ser seleccionada sobre tres principales restricciones:

1. Un amplio conjunto de funciones de aproximación, $\mathscr{LM}$, que definirá el *espacio de aproximación.*
2. Un número limitado de ejemplos, $\mathscr{T}$, denominado *conjunto de entrenamiento.*

3. Una medida de discrepancia, $L(y, f(x, \omega))$, entre la respuesta del sistema S y la respuesta de la máquina de aprendizaje LM.

Mientras que la segunda restricción viene determinada por el problema concreto a tratar, sobre la primera es necesaria la intervención del investigador, puesto que depende en esencia del tipo de máquina de aprendizaje seleccionado. En cuanto a la tercera restricción, su uso en la definición teórica del funcional de riesgo no resulta dificultosa. Sin embargo, la imposibilidad de ser tratado de forma práctica debido al desconocimiento de la función de densidad de probabilidad conjunta, obliga a su substitución por alguna otra medida que dependerá del principio inductivo seleccionado por el investigador y el conocimiento a priori que se desee insertar para conseguir la unicidad en la solución.

2.3 Casos Particulares del PGAE

El PGAE puede ser dividido en diferentes tipos según sea la naturaleza de las entradas y salidas tratadas en el modelo general. Puesto que las entradas han sido restringidas en la definición inicial al caso real multivariable, $x \in X \subseteq \mathbb{R}^d$, las posibles variaciones del problema general vendrán determinadas por la topología de las variables de salida, $y \in Y$, siendo los dos tipos más comunes de variable: la numérica y la categórica.

2.3.1 Estimación de Regresiones

Se entenderá por *variable numérica* aquella cuyo valor pertenece a un conjunto sobre el que ha sido definida una relación de orden total, es decir, la relación de orden permite definir una distancia. El ejemplo más usual de variable numérica es el de las variables definidas sobre la recta real, $y \in Y \subseteq \mathbb{R}$. En este caso, la ordenación sobre la recta permite definir una distancia, como por ejemplo la distancia euclídea.

Definición 2.3.1. *Se define el Problema General de Estimación de una Regresión como el PGAE en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un subconjunto de la recta real, $y \in Y \subseteq \mathbb{R}$.*

La forma que adopta la función de coste para el caso de estimación de regresiones viene establecida por la siguiente afirmación.

Proposición 2.3.2. *Sean $y \in Y \subseteq \mathbb{R}$ el tipo de respuesta de un sistema y $\mathscr{L}\mathscr{M}$ un conjunto aproximación basado en funciones reales definido en 2.2.1. Sea la función de regresión definida como*

$$f^{opt}(x) = \int y p(y|x), \tag{2.3.1}$$

entonces, si la expresión del funcional de riesgo definida en (2.2.3) *es utilizada la función de coste*

$$L(y, f(x,\omega)) = (y - f(x,\omega))^2, \tag{2.3.2}$$

se puede afirmar que

- Si $f^{opt}(x) \in \mathscr{L}\mathscr{M} \Rightarrow \exists \omega^{\mathscr{L}\mathscr{M}} \in \Omega, f(x,\omega^{\mathscr{L}\mathscr{M}}) = f^{opt}(x)$.
- Si $f^{opt}(x) \notin \mathscr{L}\mathscr{M} \Rightarrow f(x,\omega^{\mathscr{L}\mathscr{M}}) = \underset{f\in\mathscr{L}\mathscr{M}}{arg\,min} \sqrt{\int (f(x,\omega) - f^{opt}(x))^2 p(x)}$.

Por lo tanto, de forma teórica, siempre es posible hallar la función de regresión solución, si ésta pertenece al conjunto de funciones que es capaz de implementar la máquina de aprendizaje o, en el caso contrario que no pertenezca, la función del espacio $\mathscr{L}\mathscr{M}$ que más se aproxima a la norma 2. A este error en la búsqueda de la solución se le denomina *error de aproximación*, por ser consecuencia de la elección del espacio de aproximación $\mathscr{L}\mathscr{M}$.

2.3.2 Reconocimiento de Patrones

Una *variable categórica* es aquella que cuyo valor pertenece a un conjunto finito sobre el que no ha sido definido una relación de orden. Ejemplos de variables categóricas pueden ser la que toman valor sobre conjunto de elementos no numéricos — colores, marcas, tipos entre otros —.

Definición 2.3.3. *Se define el problema general de clasificación o de reconocimiento de patrones como el PGAE en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un conjunto cuyos elementos no están ordenados,* $y \in Y = \{\theta_1, \ldots, \theta_k\}$.

Los elementos del conjunto de salida reciben el nombre de *etiqueta* definiendo la *clase* a la que puede ser asignado un elemento de entrada al sistema. Es muy habitual hallar problemas de aprendizaje denominados *dicotomías*, cuyo conjunto de salida categórico ha sido transformado en variables binarias numéricas, en general $\{0,1\}$. La razón principal estriba en la factibilidad de definir una función de coste que contabilice el número de errores de clasificación.

Sean $y \in Y = \{0,1\}$ la respuesta del sistema y $\mathscr{L}\mathscr{M}$ un conjunto de aproximación basado en funciones indicador — funciones que sólo toman dos valores: cero y uno —. Es posible contar el número de errores de clasificación si en la expresión del funcional de riesgo (2.2.3) es utilizada la función de coste

$$L(y, f(x,\omega)) = \begin{cases} 0 & si\, y = f(x,\omega) \\ 1 & si\, y \neq f(x,\omega) \end{cases}, \tag{2.3.3}$$

por lo que se puede establecer la siguiente definición.

Definición 2.3.4. *Se define el Problema de Clasificación Binaria como el problema general de clasificación en el caso que el espacio de salida del sistema y de la máquina de aprendizaje sea el conjunto de etiquetas* $y \in Y = \{0,1\}$ *y funcional de riesgo* (2.2.3) *a minimizar tenga por función de coste la expresión* (2.3.3).

2.3.3 Regresión Ordinal

Además de los dos tipos generales de variables ya expresados — numérica y categórica —, es posible definir un tipo intermedio de variables denominado ordinales, que reune características de las dos clases.

Una *variable ordinal* es aquella cuyo valor pertenece a un conjunto finito de etiquetas sobre el que ha sido definida una relación de orden u ordenación. Como ejemplo de variable ordinal puede ser un conjunto de etiquetas definiendo las notas de los estudiantes en una asignatura, o bien colores que han sido ordenados por el nivel de rojo.

Definición 2.3.5. *Se define el Problema General de Regresión Ordinal o de ordenación como el PGAE en la caso que el espacio de salida del sistema y de la máquina de aprendizaje sea un conjunto finito cuyos elementos poseen una ordenación,* $y \in Y = \{\theta_1, \ldots, \theta_k\}$ *con* $\theta_k \succ \theta_{k+1} \succ \cdots \succ \theta_1$.

Se entenderá que la ordenación sobre las salidas permite establecer una ordenación sobre las entradas que sean de utilidad. Por ejemplo, un producto preferido a otro en el caso que su salida nivel de calidad sea superior respecto al orden $\succ$.

2.4 Algoritmo de Aprendizaje

Un algoritmo de aprendizaje será aquel proceso capaz de dar respuesta al problema de aprendizaje a partir de ejemplos planteado. Continuando la definición de este problema, se sucede la siguiente definición, en congruencia con aquella planteada por los autores [Vapnik, 1998]y [Cherkassky and Mulier, 1998].

Definición 2.4.1. *Se define Algoritmo de Aprendizaje a partir de ejemplos como aquel proceso capaz de elegir una única función a partir del conjunto de entrenamiento dando respuesta al problema planteado de aprendizaje a partir de ejemplos.*

Un algoritmo de aprendizaje precisa seleccionar:

1. Un *espacio de aproximación* $\mathscr{L}\mathscr{M}$ amplio y flexible.
2. Un *conocimiento a priori* que establezca una ordenación de las funciones de aproximación de acuerdo a alguna medida de flexibilidad para adecuarse a los datos del conjunto de entrenamiento.

3. Un *principio inductivo* que determine en qué medida se combina el conocimiento a priori con el conjunto de entrenamiento disponible.

4. Un *método de aprendizaje* que constituya una implementación computacional constructiva de un principio inductivo para un espacio de aproximación dado.

Los elementos iniciales a seleccionar dependen generalmente de la elección que realice el investigador para el problema concreto que está trabajando, aunque la necesidad de traducir el conocimiento a priori en términos de la metodología algorítmica empleada, restringen esta segunda elección y la estandarizan de gran manera.

2.4.1 Espacio de Aproximación

Definición 2.4.2. *La Amplitud de un Espacio de Aproximación se define como la capacidad para aproximar cualquier función continua, $f \in \mathscr{C}(X,Y)$, con una precisión especificada cualquiera. Se dirá que un espacio de aproximación $\mathscr{LM}$ es denso, en $\mathscr{C}(X,Y)$, si cumple la propiedad de aproximación universal.*

Tradicionalmente, han sido considerados buenos espacios de aproximación aquellos que son densos. Sin embargo, esta gran capacidad de aproximación provoca que el número de funciones solución que se ajustan al conjunto de entrenamiento sea muy elevado y de características muy diferentes.

Definición 2.4.3. *La Flexibilidad de un Espacio de Aproximación se define como la capacidad del espacio para estimar dependencias arbitrarias a partir de un conjunto de datos finitos.*

La flexibilidad es una propiedad que depende en gran medida del tipo de máquina de aprendizaje definitoria del espacio de aproximación y su habilidad para trabajar sobre un conjunto finito de datos.

2.4.2 Conocimiento a Priori

La necesidad de asumir cierto conocimiento de antemano sobre la forma del modelo buscado es esencial para conseguir la unicidad de la solución. Tal conocimiento será insertado en el algoritmo de aprendizaje en función del principio inductivo seleccionado. Algunos de los requerimientos más comunes suelen ser la exigencia a la suavidad en la solución, la reducción en la talla de los pesos, la existencia de una determinada función de probabilidad conocida, la maximización del margen entre clases, entre otras.

2.4.3 Principios Inductivos

Observando el funcional de riesgo (2.2.3) a ser minimizado según la definición del PGAE, se deduce que la función desconocida debe ser estimada para todo posible valor del espacio de entrada X, ya que la esperanza es tomada sobre alguna distribución de probabilidad desconocida del espacio completo. La función solución no sólo debe tener un buen comportamiento sobre el conjunto de entrenamiento, sino que deber ser asegurado que posea la propiedad de *generalizar* bien, por lo que se hace necesario estimar un modelo general que permita predecir la salida a cualquier entrada del espacio X, siguiendo el proceso de inferencia inductiva o de inducción-deducción, que se puede ver en la Figura (2.2). En paralelo, la misma definición restringe la creación de un algoritmo a la de un espacio aproximador definido — aunque tan amplio y flexible como se desee —, un conjunto de entranamiento finito no controlable y una medida de discrepancia basada en una distribución de probabilidad desconocida. El principio inductivo deberá establecer cómo dar respuesta al problema de aprendizaje generalizando bien y cumpliendo las restricciones del planteamiento.

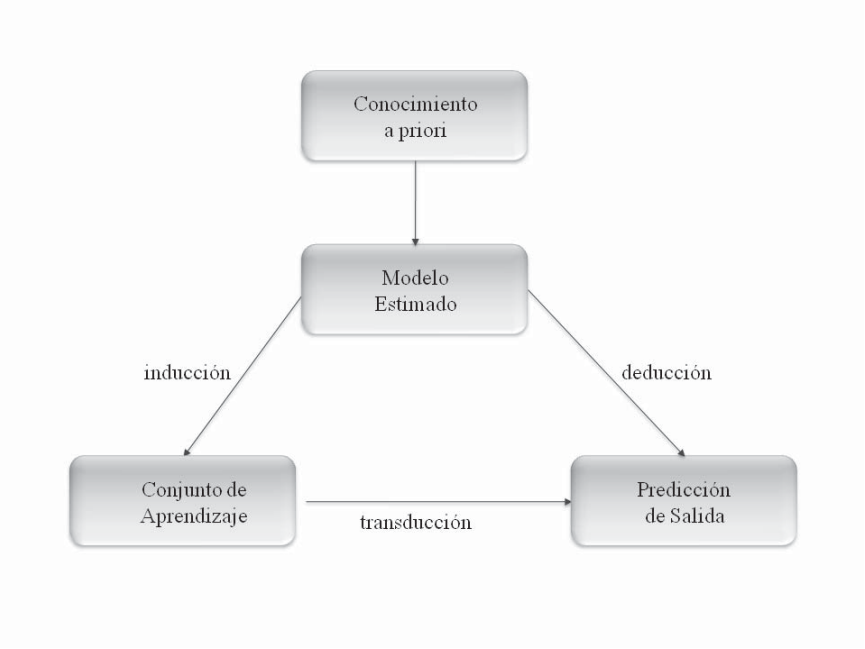


Figura 2.2: *Procesos de Inferencia: inducción-deducción y transducción.*

Minimización del Riesgo Empírico

El principio empírico de minimización del riesgo empírico — ERM, del inglés *Empirical Risk Minimization* — es el más comúnmente utilizado en los procesos de aprendizaje clásico. Tomando como espacio de aproximación alguno que sea denso, por lo que queda eliminada la primera de las tres restricciones impuestas por la definición del PGAE. De las otras dos, puesto que el conjunto de entrenamiento no es controlable, el

principio ERM fija su atención en redefinir el funcional de riesgo basándose en el siguiente razonamiento.

> "Para obtener una buena generalización es suficiente con elegir los parámetros de la función aproximadora que aseguren el número mínimo de errores sobre el conjunto de entrenamiento".

Siguiendo esta aseveración, el principio inductivo ERM sustituye la minimización del funcional de riesgo (2.2.3) con función de densidad desconocida por el siguiente esquema:

1. El funcional de riesgo $R(\omega)$ es reemplazado por el denominado *funcional de riesgo empírico*

$$R_{emp}(\omega) = \frac{1}{\ell}\sum_{p=1}^{\ell} L(y_p, f(x_p, \omega)), \tag{2.4.1}$$

   construido sobre el conjunto de entrenamiento $\mathscr{T}$. Generalmente la función de coste sigue siendo la expresión (2.3.2)de la norma 2 al cuadrado.

2. Se aproxima la función (2.2.4), $f(x, \omega^{\mathscr{L},\mathscr{M}})$, que minimiza el riesgo (2.2.3) por la función $f(x, \omega^{\mathscr{T}})$ que minimiza el riesgo (2.4.1).

Definición 2.4.4. *[Vapnik, 1995]. Se define el Error de Generalización cometido por un algoritmo de aprendizaje al seleccionar un problema de aprendizaje como la cota de la suma del error de aproximación que se comete al elegir el espacio de aproximación más el error de estimación que provoca la finitud del conjunto de entrenamiento en el que se base el proceso de aprendizaje.*

Siguiendo la definición, se dirá que la función solución estimada generaliza bien si comete un error de generalización pequeño. Nuevamente esta medida es puramente teórica e imposible de calcular, pero permite definir una base teórica sobre la que definir principios de inferencia que aseguren una buena generalización.

El uso del principio ERM en procesos de aprendizaje clásicos, conduce a un proceso de interpolación sobre el conjunto de entrenamiento que conlleva al fenómeno conocido como *sobre-entrenamiento* u *overfitting*, que habitualmente es solucionado mediante el uso de criterios de parada temprana del entrenamiento — en inglés,. *Early Stopping Rules* — o mediante el decaimiento de pesos — en inglés, *Weigth Decay* — [Bichop, 1995]. Estas técnicas son una introducción de nueva información a priori durante el proceso de aprendizaje, que resulta del todo necesaria para asegurar la unicidad de la solución.

Regularización o Penalización

El proceso de inducción de un modelo general a partir de un conjunto de entrenamiento en un problema mas situado en el sentido que no existe una única solución.

La técnica de regularización [Tikhonov and Arsenin, 1977] asegura, bajo ciertas pequeñas restricciones sobre los espacios de trabajo, que si en vez del funcional de riesgo $R(\omega)$ se minimiza el denominado funcional de riesgo regularizado

$$R^{reg}(\omega) = R(\omega) + \lambda \cdot \phi(f), \tag{2.4.2}$$

donde $\phi(f)$ es algún tipo de funcional y λ es una constante positiva escogida apropiadamente, entonces se obtiene una única solución sobre el espacio definido por el funcional $\phi(f)$.

Básicamente se trata de traducir las condiciones a priori que se quieren y/o se deben imponer para conseguir solución única en la forma de un funcional penalizador. Este método de inferencia inductiva, al ser combinado con el principio ERM, permite definir el *funcional de riesgo empírico regularizado* a ser minimizado para solucionar el problema de aprendizaje como

$$R^{reg}_{emp} = R_{emp}(\omega) + \lambda \cdot \phi(f), \tag{2.4.3}$$

demostrando que el razonamiento sobre el que se basa el método ERM de minimizar $R_{emp}(\omega)$ para asegurar una buena generalización.

Inferencia Bayesiana

La inferencia bayesiana codifica información a priori adicional sobre las funciones de aproximación en forma de una distribución de probabilidad a priori, la probabilidad de que una función del espacio $\mathscr{L}\mathscr{M}$ sea la auténtica función desconocida. De esta forma responde a la restricción sobre la definición del funcional de riesgo (2.2.3) asumiendo una cierta probabilidad sobre el modelo y añadiendo información con intención de obtener un único modelo predictivo.

Este tipo de inferencia está basada en la fórmula clásica de Bayes de actualización de probabilidades a priori utilizando la evidencia proporcionada por los datos

$$P[modelo|datos] = \frac{P[datos|modelo] \cdot P[modelo]}{P[datos]}, \tag{2.4.4}$$

donde $P[modelo|datos]$ es la probabilidad a posteriori que se desea conocer y $P[modelo|datos]$ es ka probabilidad a priori usada antes de que la máquina de aprendizaje observe los datos.

El gran inconveniente de la técnica bayesiana es su restricción a la necesidad de que el conjunto de aproximación $\mathscr{L}\mathscr{M}$ coincida con el conjunto de problemas que la máquina

tiene que resolver. Tal como se ejemplariza en [Vapnik, 1995], no tiene sentido aplicar inferencia bayesiana a un problema de aproximación por polinomios si la función regresión no es polinómica, puesto que la probabilidad a priori de que cualquier función de $\mathscr{LM}$ sea la función de regresión igual a 0.

Minimización del Riesgo Estructural

El principio inductivo de *Minimización del Riesgo Estructural* — SRM, del inglés *Structural Risk Minimization* — es un proceso de inferencia desarrollado sobre la *Teoría del Aprendizaje Estadístico* — SLT, del inglés *Statiscal Learning Theory* [Vapnik, 1998]— específicamente para trabajar con problemas de aprendizaje a partir de un conjunto de entrenamiento pequeño.

A partir de la obtención de una cota sobre el funcional empírico $R(\omega)$ válida para cualquier conjunto dado de funciones $\mathscr{LM}$, se concluye que para asegurar su minimización, fijado el conjunto de entrenamiento, es necesario minimizar simultáneamente el funcional de riesgo empírico $R_{emp}(\omega)$ y la VC dimensión – del inglés, Vapnik–Chervonenkis Dimension — del conjunto de funciones $\mathscr{LM}$, $\Phi(h_K,\ell)^2$, que es una medida de la amplitud del espacio de aproximación.

$$R(\omega) \leq R_{emp}(\omega) - \Phi(h_K,\ell) = R_{srm}(\omega). \tag{2.4.5}$$

Para entender mejor esta medida de la amplitud del espacio de aproximación, sirva la siguiente definición restringida para el caso de un problema de clasificación binaria.

Definición 2.4.5. *(Vapnik-Chervonenkis). La VC dimensión de un conjunto de funciones indicador $\mathscr{LM}$ es igual al número h máximo de vectores $(x_1,y_1),\ldots,(x_\ell,y_\ell)$ que pueden ser separados en dos clases diferentes en todas las 2^h posibles maneras utilizando este conjunto de funciones $\mathscr{LM}$.*

Corolario 2.4.6. *(Vapnik-Chervonenkis). Si el espacio de aproximación $\mathscr{LM}$ es denso, entonces la VC dimensión es infinita, $h=\infty$, por lo que la minimización del riesgo empírico no se puede asegurar, siguiendo la expresión* (2.4.5)*, la minimización del riesgo funcional, $R(\omega) \leq R_{emp}(\omega) + \infty = \infty$.*

Definición 2.4.7. *Se dirá que un espacio de aproximación $\mathscr{LM}$ es capaz si permite asegurar la minimización del funcional de riesgo $R_{srm}(\omega)$.*

El funcional de riesgo estructural constituye una cota del funcional de riesgo, por lo que se trata de una condición suficiente para asegurar la minimización de $R(\omega)$, pero en ningún caso es una condición necesaria. Siempre es posible en casos de trabajo sobre problemas reales emplear un espacio de aproximación denso aunque se aplique el principio inductivo SRM y obtener buenos resultados, sin que la cota del riesgo funcional pueda asegurar de antemano estos buenos resultados.

2.4.4 Métodos de Aprendizaje

Un método de aprendizaje resulta de la implementación constructiva del principio inductivo elegido en el algoritmo de aprendizaje. Generalmente corresponde a un proceso de optimización de un cierto funcional de riesgo que ha sido determinado siguiendo el principio inductivo.

Para cada principio inductivo existen muchos métodos de aprendizaje que lo implementan, que corresponden a los diferentes espacios de aproximación $\mathscr{L}\mathscr{M}$ y a las diferentes técnicas de optimización.

A continuación se mencionarán algunos métodos generales que se utilizan sobre los principios inductivos de regulariazación y SRM.

Métodos sobre el Principio de Regularización

Aunque el método inicial de minimizar el funcional de riesgo empírico regularizado (2.4.3), que constituye al funcional de riesgo teórico $R(\omega)$, es habitualmente solucionado en su formulación original, también podría ser expresado como

$$min\, R_{emp}(\omega)\ con\ \phi(f) < C, \tag{2.4.6}$$

es decir, aplicar el principio ERM mientras se mantiene la complejidad del modelo acotada mediante el término de regularización $\phi(f)$.

Otro método podría plantearse sin el problema inicial es desarrollado como

$$min\, \phi(f)\ con\ R_{emp}(\omega) < \delta. \tag{2.4.7}$$

Todos estos enunciados corresponden a problemas de optimización convexa que pueden ser solucionados eficientemente y son equivalentes [Smola, 1998].

Métodos sobre el Principio SRM

Para solucionar el problema de minimización de la cota de riesgo estructural (2.4.5) asociado al principio SRM, es necesario hallar un resultado que evite el overfitting y que minimice el número de errores sobre $\mathscr{T}$. Los métodos de aprendizaje pueden realizar la tarea siguiendo una de estas dos visiones:

1. Elegir una arquitectura apropiada para la máquina de aprendizaje que permita mantener la amplitud del espacio de aproximación acotada y minimizar el riesgo

empírico

$$min\, R_{emp}(\omega)\ con\ \phi(h_k, \ell) < C. \tag{2.4.8}$$

2. Mantener el valor del funcional de riesgo empírico fijado — por ejemplo igual a 0 — y minimizar la amplitud de $\mathscr{L}\mathscr{M}$

$$min\, \phi(h_k, \ell)\ con\ R_{emp}(\omega) < \delta. \tag{2.4.9}$$

Según se tenga en consideración una u otra expresión del problema es posible implementar dos tipos de máquinas de aprendizajes diferentes:

1. Redes Neuronales Artificiales — ANN, del inglés *Artificial Neural Networks* — .
2. Máquinas de Soporte Vectorial — SVM, del inglés *Support Vector Machines* —.

Capítulo 3

Máquinas de Soporte Vectorial

3.1 Introducción

El problema de modelado de datos empíricos es pertinente para muchas aplicaciones de la ingeniería. Estos datos son modelados a través de un proceso de inducción, que es utilizado para construir un modelo que defina el sistema de los datos, del cual se espera deducir las respuestas a eventos que aún no han sido observados en el pasado. Fundamentalmente la calidad y la cantidad de las observaciones, van a establecer el desempeño del modelo empírico.

El hecho de que los datos sean finitos y muestreados — un subconjunto de casos o individuos de una población —, hacen que generalmente la muestra no sea uniforme, ya que la naturaleza dimensional de los datos del problema es sólo una distribución dispersa en el espacio de entrada. Por lo tanto, el problema es casi siempre mal definido.

Los enfoques tradicionales de *Redes Neuronales* han sufrido dificultades con la generalización, teniendo así una gran susceptibilidad a producir modelos altamente sobreajustados a los datos — denomidado *overfitting* —. Esto es una consecuencia de los algoritmos de optimización utilizados para la selección de los parámetros y las medidas estadísticas para seleccionar el mejor modelo [T. Poggio and Koch., 1985].

Los cimientos de *Máquinas de Soporte Vectorial* — SVM del inglés, *Support Vector Machines* — han sido desarrollados por [Vapnik, 1995] y están ganando popularidad debido a un conjunto atractivo de características y comportamiento empírico muy prometedor. La formulación encarna el principio de *Minimización del Riesgo Estructural* — definido en la página 14—, que ha demostrado ser superior al ya tradicional principio de *Minimización del Riesgo Empírico* [S.R. Gunn and Bossley, 1997] — definido en la página 11—, empleado por las redes neuronales convencionales. Es esta diferencia la que equipa a las SVM con mayor capacidad de generalizar, que es el objetivo del aprendizaje estadístico. Las SVM se desarrollaron para resolver el problema de clasificación, pero recientemente se han ampliado al ámbito de los problemas de regresión [Vapnik and Smola, 1997].

Utilizando el principio de inducción de Minimización del Riesgo Estructural, SRM, como proceso de inferencia, se desea construir un método de aprendizaje que permita dar respuesta al *Problema General de Aprendizaje a partir de Ejemplos* — definido en la página 6—. Debido a que el principio SRM se basa en un proceso de construcción de la solución sobre espacios anidados cuya amplitud o capacidad está determinada por funciones indicador, el tipo de problema de aprendizaje para el que resulta más sencillo hallar un método es el de clasificación binaria mediante hiperplanos — método SVMC, del inglés *Support Vector Machine for Classification* —, pero que gracias a la adición de un nuevo componente denominado función de pérdida o *loss function*, también resulta viable para problemas de regresión — Método SVMR, del inglés *Support Vector Machine for Regression* —. Cabe destacar que el enfoque con el cual este trabajo estará fundamentado será sobre este último método.

3.2 Hiperplano Óptimo de Separación

Considere el problema de separar el conjunto de Datos de Entrenamiento — $\mathscr{D}$, del inglés *Dataset* — en dos clases

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}, x \in X \subseteq \mathbb{R}^d, y \in Y = \{-1, 1\}, \tag{3.2.1}$$

con un hiperplano

$$\langle \omega, x \rangle + b = 0 \tag{3.2.2}$$

Definición 3.2.1. *Un conjunto de vectores está Óptimamente Separado por un hiperplano, si éste es separado sin error y la distancia entre el vector más cercano al hiperplano es máxima.*

Para evitar la pérdida de generalidad, resulta apropiado considerar la forma canónica del hiperplano [Vapnik, 1995], donde los parámetros ω, b se ven limitados por la siguiente restricción

$$\min_i |\langle \omega, x_i \rangle + b| = 1, \tag{3.2.3}$$

La restricción (3.2.3) establece que: *la norma del vector de pesos debe ser igual a la inversa de la distancia, del punto más cercano en el conjunto de datos al hiperplano.* Esta idea se ilustra en la Figura 3.1, donde se muestra la distancia del punto más cercano a cada hiperplano.

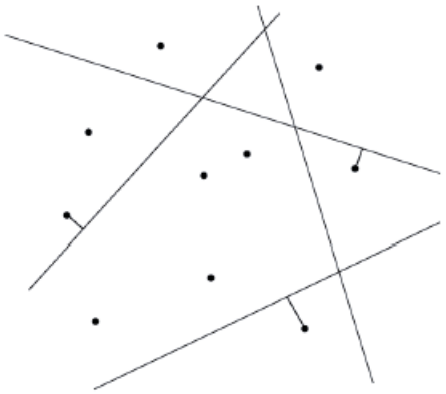


Figura 3.1: *Hiperplanos Canónicos*

El hiperplano separador — en su forma canónica — debe satisfacer la siguiente restricción

$$y_i\left[\langle \omega, x_i\rangle + b\right] \geq 1,\ i = 1,\ldots,\ell. \tag{3.2.4}$$

La distancia $d(\omega, b; x)$ de un punto x al hiperplano (ω, b) está definido por

$$d(\omega, b; x) = \frac{|\langle \omega, x\rangle + b|}{\|\omega\|}. \tag{3.2.5}$$

El hiperplano óptimo viene dado por la maximización del margen, ρ — definido como dos veces la distancia desde el hiperplano hasta el (o los) ejemplo(s) mas cercano(s) a este —, sujeto a las limitaciones de la ecuación 3.2.4. El margen está dado por

$$\begin{aligned}
\rho(\omega, b) &= \min_{x_i : y_i = -1} d(\omega, b; x_i) + \min_{x_i : y_i = 1} d(\omega, b; x_i) \\
&= \min_{x_i : y_i = -1} \frac{|\langle \omega, x_i\rangle + b|}{\|\omega\|} + \min_{x_i : y_i = 1} \frac{|\langle \omega, x_i\rangle + b|}{\|\omega\|} \\
&= \frac{1}{\|\omega\|}\left(\min_{x_i : y_i = -1} |\langle \omega, x_i\rangle + b| + \min_{x_i : y_i = 1} |\langle \omega, x_i\rangle + b|\right) \\
&= \frac{2}{\|\omega\|}.
\end{aligned} \tag{3.2.6}$$

Por lo tanto el hiperplano que separa los datos de forma óptima, es el que minimiza $\phi(\omega)$

$$\phi(\omega) = \frac{1}{2}\|\omega\|^2. \tag{3.2.7}$$

Corolario 3.2.2. *El Hiperplano Óptimo de Separación — OSH, del inglés Optimum Separation Hyperplane — es un clasificador lineal con el margen máximo para un determinado conjunto finito de patrones de aprendizaje. El cálculo del OSH con una máquina de soporte vectorial lineal viene dado por el siguiente problema de optimización*

$$\begin{aligned} & min\, \phi(\omega) = \tfrac{1}{2}\|\omega\|^2 \\ s.a. & \\ & y_i\left[\langle \omega, x_i\rangle + b\right] \geq 1, i = 1,\ldots,\ell \end{aligned} \tag{3.2.8}$$

3.3 SVM para Clasificación

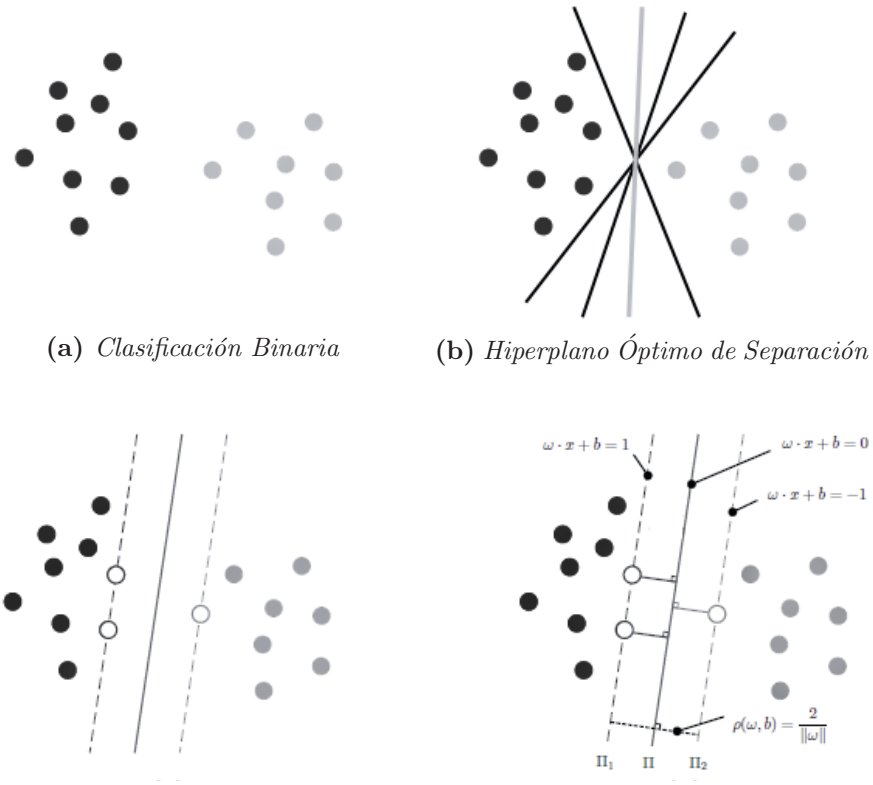
El problema de clasificación se puede limitar a la consideración del problema de dos clases — problema dicotómico abordado anteriormente en la en la página 8— como lo muestra la Figura 3.2a, sin pérdida de generalidad. En este problema, el objetivo es separar las dos clases por una función que se induce a partir de los ejemplos disponibles. El objetivo es producir un clasificador que funcione bien sobre los ejemplos que no ha podido visualizar, es decir, que tenga un alto grado de generalización. [Smola, 1998],[Vapnik, 1995]y [Burges, 1998].

Consideremos el ejemplo de la Figura 3.2b. Aquí hay muchos posibles clasificadores lineales que pueden separar los datos, pero sólo hay uno que maximiza el margen — maximiza la distancia entre éste y el punto más cercano de datos de cada clase —. Este clasificador lineal se denomina *Hiperplano Óptimo de Separación* — obtenido a través del problema de optimización definido en (3.2.8)—. Intuitivamente, podríamos esperar que este límite generalice mejor que las otras fronteras posibles.

Definición 3.3.1. *Supóngase que se dispone de un conjunto de ejemplos, todos pertenecientes a un espacio característico, algunos de los cuales pertenecen a la clase $\mathscr{C}_-$ y los restantes, a la clase $\mathscr{C}_+$. Se define SVM para Clasificación — SVMC, del inglés Support Vector Machine for Classification — a la factibilidad de construir un hiperplano $\mathscr{H}$ — en este caso binario — $\mathscr{H} = \{x \in X : \omega \cdot x + b = 0\}$ con parámetros $\omega \in X \setminus \{0\}$ y $b \in \mathbb{R}$, tal que se cumplan estas dos condiciones:*

1. *Los ejemplos de la clase $\mathscr{C}_-$ — donde $\mathscr{C}_- = \{x \in X : \omega \cdot x + b < 0\}$ — deben quedar a un lado del hiperplano y los de la clase $\mathscr{C}_+$ — donde $\mathscr{C}_+ = \{x \in X : \omega \cdot x + b > 0\}$ — en el otro lado del hiperplano $\mathscr{H}$ — ver Figura 3.2d —.*

2. *El margen del hiperplano ρ debe ser máximo — definido en la Ecuación* (3.2.6)*, ver Figura 3.2c —.*



(a) *Clasificación Binaria* **(b)** *Hiperplano Óptimo de Separación*

(c) *Margen Óptimo* **(d)** *Elementos Hiperplano Óptimo*

Figura 3.2: *Problema de Clasificación.*

Corolario 3.3.2. *Es fácil darse cuenta de una característica muy importante de las SVMs y es que si se añade o elimina cualquier número de vectores que cumplan la desigualdad estricta, la solución del problema de optimización para el hiperplano óptimo* $\Pi = \omega \cdot x + b = 0$ *no se ve afectada. Sin embargo, basta con añadir un vector que se encuentre entre los dos hiperplanos — es decir, dentro del margen óptimo —, para que la solución cambie totalmente. De acá se desprende el elemento esencial de las SVMs, que son los Vectores Soporte — SV, del inglés Support Vectors —, que corresponden a todos los vectores que se encuentran al máximo margen* ρ *del hiperplano óptimo, es decir que se encuentran en uno de los hiperplanos* $\Pi_1 = \omega \cdot x + b - 1$ *,* $\Pi_2 = \omega \cdot x + b + 1$ *— Representados en círculos blancos en la Figura 3.2d—*

3.3.1 Modelos Lineales de SVMC

Definición 3.3.3. *Se define como Modelo Lineal de Vectores Soporte — conocido también como caso linealmente separable para SVM — a un problema de clasificación — definido en la página 8— donde es posible aplicar una SVMC.*

En función a lo anteriormente expuesto, es que será necesario resolver el problema de

optimización con restricciones del OHC. — definido en (3.2.8)— para poder dar solución al problema de clasificación.

Para resolver el problema de optimización será necesario utilizar los *Multiplicadores de Lagrange*. La función quedaría de la siguiente manera:

$$L_p(\omega, b, \alpha_i) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{\ell} \alpha_i \cdot (y_i \cdot (x_i \cdot \omega + b) - 1). \tag{3.3.1}$$

El problema queda como un problema de *Programación Cuadrática* — QP, del inglés *Quadratic Programming* — donde la función objetivo es convexa, y los vectores que satisfacen las restricciones forman un conjunto convexo. Esto significa que se puede resolver el siguiente problema dual asociado al problema primal: maximizar la función $L_p(\omega, b, \alpha_i)$ respecto a las variables duales α_i sujeta a las restricciones impuestas para que los gradientes de L_p con respecto a ω y b sean nulos, y sujeta también al conjunto de restricciones $C = \{\alpha_i \geq 0, i = 1, \ldots, \ell\}$. La solución de este problema se expresa en la forma de

$$\omega = \sum_{i=1}^{\ell} \alpha_i y_i x_i \quad , \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0 \ , \tag{3.3.2}$$

y la función objetivo dual a maximizar

$$L_d = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j x_i x_j \tag{3.3.3}$$

Los vectores del conjunto de entrenamiento que proporcionan un multiplicador $\alpha_i > 0$ son los vectores soporte. Para este tipo de modelo de aprendizaje, los vectores soporte son los elementos críticos, ya que ellos son los que proporcionan la aproximación del problema. — ver Colorario (3.3.2) —

Las condiciones del problema de optimización nos lleva a que se cumple con la condición *Karush-Kuhn-Tucker* — KKT —, que son condiciones necesarias y suficientes para que la solución sea óptima. Es una generalización del método de los *Multiplicadores de Lagrange*[Karush, 1939]. Esta es representada de la siguiente forma

$$\alpha_i \cdot (y_i \cdot (x_i \cdot \omega + b) - 1) = 0. \tag{3.3.4}$$

Estas restricciones indican que el producto de las restricciones del problema primal $(\alpha_i \cdot (y_i \cdot (x_i \cdot \omega + b) - 1) = 0)$ y las restricciones del problema dual $\alpha_i \geq 0$ se anulan en todos los vectores de entrenamiento. De esta forma se sigue que las condiciones KKT

[Fletcher, 1987], para el problema primal definido a partir de la función objetivo son las siguientes:

$$\begin{aligned}
\omega_j - \sum_{i=1}^{\ell} \alpha_i y_i x_{ij} &= 0 \qquad j = 1, \ldots, d \\
\tfrac{\partial}{\partial b} L_p = -\sum_{i=1}^{\ell} \alpha_i y_i &= 0 \\
y_i \cdot (x_i \cdot \omega + b) - 1 &\geq 0 \qquad i = 1, ..., \ell \\
\alpha_i &\geq 0 \qquad i = 1, ..., \ell \\
\alpha_i \cdot (y_i \cdot (x_i \cdot \omega + b) - 1) &= 0 \qquad i = 1, ..., \ell
\end{aligned} \tag{3.3.5}$$

De los desarrollos iniciales no se sigue una forma explícita de determinar el valor b, sin embargo, la condición KKT complementaria nos permite determinarlo. Para ello, basta elegir un $\alpha_i > 0$ y despejar el valor de b obteniendo $b = y_i - x_i \cdot \omega$. Aunque se ha determinado b, es más adecuado realizar los cálculos con todos los $\alpha_i > 0$ y elegir como valor de b un valor promedio de los resultados obtenidos, con objeto de redondear los errores intrínsecos asociados a todo método de cálculo numérico[1]

$$\begin{aligned}
b &= \left(\frac{1}{\sum_{i=1}^{\ell} fb_1(\alpha_i)}\right) \sum_{i=1}^{\ell} fb_1(\alpha_i) \cdot (y_i - x_i \cdot \omega) \\
&\textit{donde} \\
fb_1(\alpha) &= \begin{cases} 1 & si\ \alpha_i > 0 \\ 0 & e.o.c \end{cases}
\end{aligned} \tag{3.3.6}$$

Una vez obtenido el vector ω y la constante b la solución al problema de optimización se interpreta a partir de la función Θ de la siguiente forma[2]:

$$\Theta(x) = \begin{cases} 1 & si\ \Pi(x) > 0 \\ -1 & si\ \Pi(x) < 0 \end{cases} \tag{3.3.7}$$

3.3.2 Modelos No Lineales de SVMC

En la práctica no es habitual trabajar con conjuntos separables linealmente. En estos casos — ver Figura — se encuentran vectores de una clase dentro de la región

[1] $fb_1(\alpha)$, función binaria que denota a un multiplicador α su positividad. — $\alpha > 0$ —

[2] Π, denota el hiperplano óptimo.

correspondiente a los vectores de la otra clase y por tanto nunca podrán ser separados de esta clase por medio de hiperplanos lineales. En estas situaciones se dirá que el conjunto es no es *Separable linealmente*. Ante estos casos, el problema de optimización del hiperplano óptimo — definido en (3.2.8) —, no encuentra una solución posible y ello es evidente sin mas que observar como la función objetivo (3.3.3) crece de forma arbitraria ya que el multiplicador de Lagrange correspondiente a este vector se puede tomar arbitrariamente un valor muy grande sin que viole las restricciones. Sin embargo, no es difícil ampliar las ideas generales del caso de clasificación lineal al caso no lineal introduciendo una variable ξ de *holgura* en las restricciones y plantear un nuevo conjunto de restricciones:

$$\begin{aligned} x_i \cdot \omega + b &\geq +1 + \xi_i \quad para\, y_i = +1 \\ x_i \cdot \omega + b &\leq -1 + \xi_i \quad para\, y_i = -1 \\ \xi_i &\geqslant 0 \qquad\qquad i = 1, ..., \ell \end{aligned} \tag{3.3.8}$$

Se tiene ahora que para que se produzca un error en la clasificación de un vector de entrenamiento — una entrada no es ubicada en la clase correcta — es necesario que el valor correspondiente a ξ_i sea superior a la unidad.

Así, si en el vector x_i se comete un error entonces $\xi_i \geq 1$ y por tanto $\sum \xi_i$ es una cota superior del número de errores que se cometen dentro del conjunto de entrenamiento. Ya que en el caso no lineal, necesariamente se han de cometer errores, parece natural asignar a la función objetivo un coste extra que penalice los errores — función de pérdida —. Por todo ello, una opción lógica será plantear el problema de minimizar

$$\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{\ell} \xi_i^i. \tag{3.3.9}$$

Si se considera un valor C grande, significa que el investigador está asignando un peso a los errores muy alto frente a $\|\omega\|^2$, y por el contrario si C es pequeño asigna un mayor peso a $\|\omega\|^2$. Esta interpretación resulta más intuitiva si se interpreta que $\|\omega\|^2$ es un factor de suavizamiento de la solución buscada. Por otro lado si k es grande lo que hacemos es dar mucho mas peso a los errores cuantos mayores sean estos. Se llega por tanto a plantear un problema de programación convexa para cualquier valor de k. En el presente trabajo se considera $k = 1$ ya que en este caso se tiene la ventaja de que ningún valor ξ_i, ni ninguno de sus correspondientes multiplicadores de Lagrange, aparecen en el problema dual. Por tanto el problema de optimización que se plantea es:

$$
\begin{aligned}
& \min \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{\ell}\xi_i. \\
s.a. \quad & \\
& y_i(x_i\cdot\omega + b) \geq 1-\xi_i, \quad i=1,\ldots,\ell. \\
& \\
& \xi_i \geq 0, \qquad\qquad\qquad i=1,\ldots,\ell.
\end{aligned}
\tag{3.3.10}
$$

Utilizando la técnica de los multiplicadores de Lagrange se llega a la función objetivo dual

$$
L_d = \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j x_i x_j \tag{3.3.11}
$$

la cual hay que maximizar respecto α_i sujeta a

$$
0 \leq \alpha_i \leq C,\, i=1,\ldots,\ell \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0 \tag{3.3.12}
$$

y cuya solución final viene dada por

$$
\omega = \sum_{i=1}^{n}\alpha_i y_i s_i \tag{3.3.13}
$$

donde n representa el número de SVs y por s_i los vectores soporte del conjunto $\{x_i,\ldots,x_\ell\}$. Claramente $n<\ell$ y una de las características más interesante de estos modelos es que eligiendo adecuadamente los parámetros es posible conseguir que n sea muy inferior, con lo que se consigue una representación mas simple de la solución en función de los vectores de entrada sin perder capacidad de generalización. Nótese, que la única diferencia en la solución con respecto a la dada en el caso lineal es que los multiplicadores de Lagrange α_i, están acotados superiormente por la constante C.

Las condiciones de Karush-Kuhn-Tucker — KKT — asociada a este problema son las siguientes

$$
\begin{aligned}
\tfrac{\partial}{\partial \omega_j} L_p = \omega_j - \sum_{i=1}^{\ell} \alpha_i y_i x_{ij} &= 0 \qquad j = 1, \ldots, d \\
\tfrac{\partial}{\partial b} L_p = -\sum_{i=1}^{\ell} \alpha_i y_i &= 0 \\
\tfrac{\partial}{\partial \xi_i} L_p = C - \alpha_i - \mu_i &= 0 \\
y_i \cdot (x_i \cdot \omega + b) - 1 + \xi_i &\geq 0 \qquad i = 1, ..., \ell \\
\xi_i, \alpha_i, \mu_i &\geq 0 \qquad i = 1, ..., \ell \\
\alpha_i \cdot (y_i \cdot (x_i \cdot \omega + b) - 1 + \xi_i) &= 0 \qquad i = 1, ..., \ell \\
\mu_i \cdot \xi_i &= 0 \qquad i = 1, ..., \ell
\end{aligned}
\tag{3.3.14}
$$

Como se comentó en el caso lineal, se pueden usar las condiciones complementarias de KKT — las ecuaciones sexta y séptima de las condiciones KKT en (3.3.14)— para determinar el valor de b. Nótese que la ecuación tercera combinada con la séptima de (3.3.14) muestra que si $\xi_i = 0$ entonces $\alpha_i < C$. Así se puede simplificar el cálculo de b tomando vectores de entranamiento tales que $0 < \alpha_i < C$ con $\xi_i = 0$. La solución al problema de optimización se interpreta a partir de la función Θ — definida en (3.3.7)— y el parámetro b dado por [3]

$$
\begin{aligned}
& b = \left(\frac{1}{\sum_{i=1}^{\ell} fb_1(\alpha_i)}\right) \sum_{i=1}^{\ell} fb_1(\alpha_i) \cdot (y_i - x_i \cdot \omega) \\
& \textit{donde} \\
& fb_1(\alpha) = \begin{cases} 1 & si\, 0 < \alpha_i < C \\ 0 & e.o.c \end{cases}
\end{aligned}
\tag{3.3.15}
$$

3.4 Espacio de Características y Funciones Kernel

En muchas ocasiones, los ejemplos que componen el Conjunto de entrada $\mathscr{T}$ viven en un espacio que no es linealmente separable, pero que guardan una estructura que perfectamente podría soportar una frontera de otro tipo. En tales casos, lo que se puede hacer es aplicar una transformación $\Phi(\cdot)$ a los ejemplos en el espacio de entrada y así trasladarlos a un espacio de mayor dimensionalidad conocido como *Espacio de Características* —$\mathscr{F}$, del inglés *Feature Space* —, en los cuales un hiperplano separador $\mathscr{H}$ sea capaz de dividir las dos clases implicadas en el problema. Utilizando esta idea,

[3] $fb_1(\alpha)$, función binaria que denota a un multiplicador α se encuentra entre 0 y C. — $0 < \alpha < C$ —

el primal del hiperplano óptimo — definido en (3.2.8)— se reformularía de la siguiente forma:

$$\begin{aligned} & min\, \phi(\omega) = \tfrac{1}{2}\|\omega\|^2 \\ s.a. \quad & \\ & y_i\left[\langle \omega, \Phi(x_i)\rangle + b\right] \geq 1,\, i = 1,\ldots,\ell \end{aligned} \tag{3.4.1}$$

Si bien, esta estrategia es viable en la mayoría de los casos, existe un método alternativo, llamado Truco del Kernel — en inglés, *Kernel Trick* — que es menos costoso y no tiene que lidiar con el problema de espacios de alta o incluso infinita dimensionalidad.

3.4.1 Función Kernel

Definición 3.4.1. *El Kernel Trick consiste en utilizar una función de Kernel que calcula directamente el producto interno entre dos vectores del espacio característico en un espacio de mayor dimensionalidad, sin necesidad de realizar el mapeo a dicho espacio. Dicha función Kernel debe necesariamente ser una función simétrica positiva semi-definida [Schölkopf and Smola., 2001].*

En la Figura (3.3) se ejemplifica gráficamente la idea del Kernel Trick.

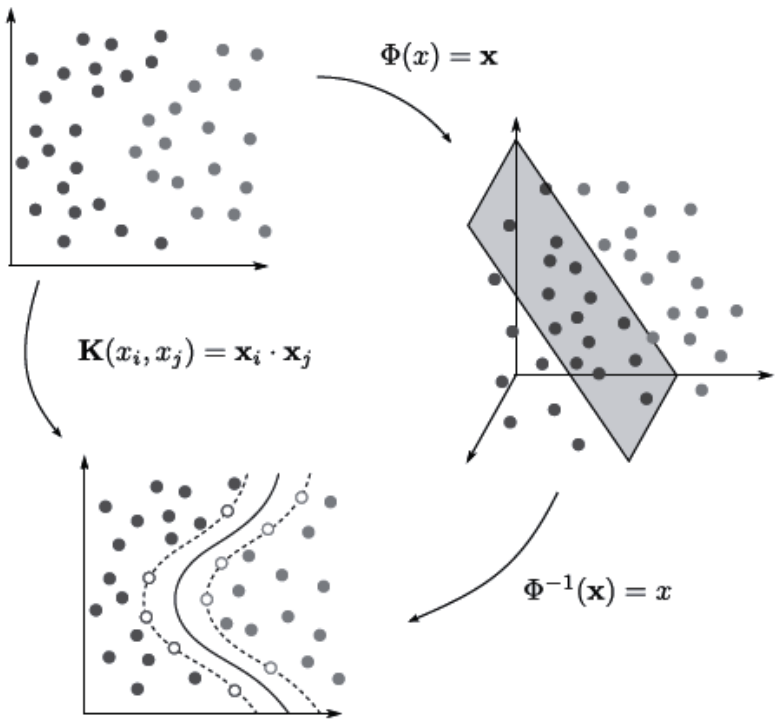


Figura 3.3: *Representación Gráfica del Kernel Trick.*

Para hacer uso de este método en las SVMs sólo se debe realizar el reemplazo de los productos puntos $\langle x_i, x_j \rangle$ por la Función de Kernel deseada $K(x_i, x_j)$ en la formulación dual para el caso de clasificación lineal — definida en (3.3.3)—:

$$\begin{aligned} L_d &= \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \\ \textit{donde} & \\ K(x_i, x_j) &= \Phi(x_i) \cdot \Phi(x_j). \end{aligned} \tag{3.4.2}$$

Desde ahora, se utilizará la función de Kernel en reemplazo del producto punto, ya que, al ser el mismo producto punto un tipo de Kernel, esta función es una forma muy conveniente y general de abarcar todos los casos. Por lo anterior, es que se hará una revisión de las funciones Kernel más utilizadas.

Polinomial

El mapeo *Polinomial* es un método popular para el modelado no lineal

$$K(x_i, x_j) = \langle x_i, x_j \rangle^d \tag{3.4.3}$$

$$K(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d \tag{3.4.4}$$

El segundo Kernel (3.4.4) es usualmente más utilizado, ya que evita los problemas ocacionados cuando el Hessiano adquiere valores 0.

Función Gaussiana de Base Radial

Las *Funciones de Base Radial* — RBF, del inglés *Radial Basis Function* — han recibido considerable atención, por lo general con una *Gaussiana* — , del inglés Gaussian Radial Basis Function — de la forma

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right). \tag{3.4.5}$$

Las técnicas clásicas que utilizan funciones de base radial emplean algún método para determinar un subconjunto de los centros. Normalmente, un método de agregación se empleó por primera vez para seleccionar un subconjunto de centros. Una característica

atractiva de la SVM es que esta selección está de manera implícita, entre los vectores de soporte que aportan una función Gaussiana local, centrada en ese punto de datos.

Función Exponencial de Base Radial

La función *Exponencial de Base Radial* produce una solución lineal a trozos que pueden ser atractivos cuando las discontinuidades son aceptables. Esta función tiene la siguiente forma

$$K(x_i, x_j) = \exp(-\frac{\left\|x_i - x_j\right\|}{2\sigma^2}). \tag{3.4.6}$$

El parámetro σ corresponde a la varianza del Kernel Guassiano.

Perceptrón Multicapa

El ya tradicional *Perceptrón Multicapa* — MLP, del inglés Multi-Layer Perceptron —, con una sola capa oculta, también tiene una representación válida para una función Kernel.

$$K(x_i, x_j) = \tanh(\rho \left\langle x_i, x_j \right\rangle + \rho) \tag{3.4.7}$$

Donde el parámetro ρ representa la *escala* o — en inglés *scale* —, mientras que ρ es el *desplazamiento* o — en inglés *offset* —. Aquí, los SVs se corresponden a la *primera capa* y los multiplicadores de Lagrange corresponden a los *pesos* — en inglés *weights* — del perceptrón.

Series de Fourier

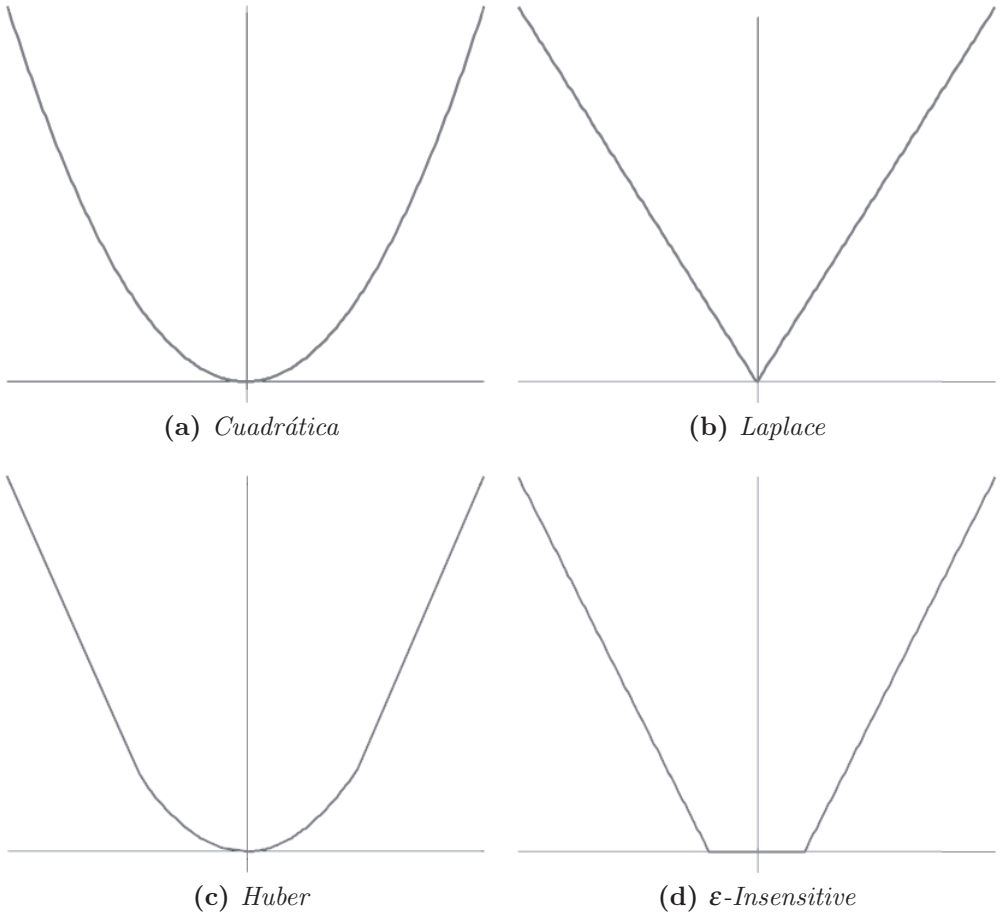
Las *Series de Fourier* son consideradas como el $2N+1$espacio característico, con respecto al espacio de entrada de dimensión N . El Kernel está definido dentro del intervalo $\left[-\frac{\pi}{2}, \frac{\pi}{2}.\right]$

$$K(x_i, x_j) = \frac{\sin(N + \frac{1}{2})(x_i - x_j)}{\sin(\frac{1}{2}(x_i - x_j))}. \tag{3.4.8}$$

Sin embargo, es probable que la elección de este Kernel no sea una muy buena opción, porque su capacidad de regularización es pobre, lo cual es evidente por la consideración de su transformada de Fourier.[Smola and Schölkopf, 1998]

3.5 SVM para Regresión

El problema de regresión — abordado anteriormente en la en la página 7 — también pueden ser solucionados a través de las SVMs, denominadas SVMR — del inglés, *Support Vector Machines for Regression* —[Cortes and Vapnik, 1995] y [Smola and Schölkopf, 1998]. Esto se logra por medio de la inclusión de una *Función de Pérdida* o — *Loss Function* en inglés — que representa el *error*. A diferencia del modelo SVMC — presentado en la página 23—, donde es básicamente un contador, en el modelo de regresión se tiene una función real. La Figura 3.4 ilustra cuatro posibles funciones de pérdida.



(a) *Cuadrática* **(b)** *Laplace*

(c) *Huber* **(d)** ε*-Insensitive*

Figura 3.4: *Funciones de Pérdida.*

La función de pérdida en la Figura 3.4a corresponde al criterio convencional del error de los mínimos cuadrados. La función de la Figura 3.4b es la función de pérdida Laplaciana, la cual es menos sensible a valores extremos que la función de pérdida cuadrática. La función de Huber propuesta en la Figura 3.4c en una función de pérdida bastante robusta ya que tiene propiedades óptimas cuando la distribución subyacente de los datos es desconocida. Estas tres funciones mencionadas anteriormente no producen

una disminución de vectores de soporte. Para solucionar este problema Vapnik propone un función de pérdida denominada ε-Insensitive — ver Figura 3.4d— que es una aproximación a la función de Huber que permite restringir el conjunto de vectores de soporte a obtener.

A continuación se presentarán los modelos lineales y no lineales para la SVMR considerando el siguiente problema de regresión

para

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}, x \in X \subseteq \mathbb{R}^d, y \in Y = \mathbb{R}, \tag{3.5.1}$$

con un hiperplano

$$\langle \omega, x \rangle + b = 0, \tag{3.5.2}$$

la función de regresión óptima viene dada por el mínimo del funcional

$$\Phi(\omega, \xi) = \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{\ell} (\xi_i^- - \xi_i^+), \tag{3.5.3}$$

donde C es un valor pre-determinado, y ξ_i^-,ξ_i^+ son variables de *holgura* que representan restricción superior e inferior respectivamente de las salidas del sistema Y.

Para ambos casos se aplicará la función de pérdida ε-Insensitive debido a las propiedades beneficiosas que otorga, la cual se define de la siguiente manera

Aplicando la función de pérdida ε-Insensitive L_ε de la Figura 3.4d tenemos lo siguiente:

$$\begin{aligned} & L_\varepsilon = \begin{cases} 0 & |f(x) - y| < \xi \\ |f(x) - y| - \xi & e.o.c \end{cases} \\ & \textit{donde} \\ & f(x) = \langle \omega, x \rangle + b \end{aligned} \tag{3.5.4}$$

3.5.1 Modelos Lineales para SVMR

Para un modelo lineal SVMR la solución está dada por

$$max W(\alpha,\alpha^*) = -\tfrac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\left\langle x_i,x_j\right\rangle + \sum_{i=1}^{\ell}\alpha_i(y_i-\varepsilon) - \alpha_i^*(y_i+\varepsilon).$$

$$\begin{aligned} &s.a \\ &\quad 0<\alpha,\alpha^*<C,\ i=1,\ldots,\ell \\ &\quad \sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)=0. \end{aligned} \tag{3.5.5}$$

Resolviendo el problema de optimización (3.5.5) se logra determinar los multiplicadores de Lagrange, α,α^*, y la función de regresión estaría dada por la Ecuación (3.5.2), donde

$$\omega = \sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)*x_i. \tag{3.5.6}$$

$$b = -\frac{1}{2}\left\langle \omega,(x_r+x_s)\right\rangle. \tag{3.5.7}$$

Las condiciones Karush-Kuhn-Tucker — KKT — que se satisfacen por la solución son:

$$\alpha_i\cdot\alpha_i^*=0,\ i=1,\ldots,\ell. \tag{3.5.8}$$

Por lo tanto los SVs son puntos en los que exactamente uno de los multiplicadores de Lagrange son mayores que cero. Cuando $\varepsilon=0$, obtenemos la función de pérdida L_1 y el problema de optimización se simplifica

$$\begin{aligned} &max W(\beta) = -\tfrac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\beta_i\beta_j\left\langle x_i,x_j\right\rangle + \sum_{i=1}^{\ell}\beta_i y_i \\ &s.a \\ &\quad -C<\beta_i<C,\ i=1,\ldots,\ell \\ &\quad \sum_{i=1}^{\ell}\beta_i=0. \end{aligned} \tag{3.5.9}$$

y la función de regresión estaría dada por la Ecuación (3.5.2) donde:

$$\omega = \sum_{i=1}^{\ell} \beta_i * x_i. \tag{3.5.10}$$

$$b = -\frac{1}{2} \langle \omega, (x_r + x_s) \rangle . \tag{3.5.11}$$

3.5.2 Modelos No Lineales para SVMR

Al igual que en problemas de clasificación, un modelo no lineal por lo general requiere adecuar el modelo de los datos. De la misma manera como el enfoque SVMC no lineal —ver en la página 23— , una asignación no lineal se puede utilizar para mapear los datos a un espacio de características de alta dimensión, donde se lleva a cabo un análisis de regresión lineal. El enfoque del Kernel es empleado para hacer frente a la complejidad de la dimensionalidad. La solución de un SVMR no lineal — utilizando la función de pérdida ε-Insensitive — está dado por:

$$\begin{aligned} & max W(\alpha, \alpha^*) = \sum_{i=1}^{\ell} \alpha_i (y_i - \varepsilon) - \alpha_i^* (y_i + \varepsilon) - \tfrac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \cdot K(x_i, x_j) \\ & s.a \\ & \quad 0 < \alpha, \alpha^* < C, \; i = 1, \ldots, \ell \\ & \quad \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0. \end{aligned} \tag{3.5.12}$$

Resolviendo el problema de optimización (3.5.12) se logra determinar los multiplicadores de Lagrange, α, α^*, y la función de regresión sería[4]:

$$f(x) = \sum_{i=1}^{n} (\alpha_i \cdot \alpha_i^*) \cdot K(x_i, x) + b. \tag{3.5.13}$$

donde

$$\langle \omega, x \rangle = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) * K(x_i, x). \tag{3.5.14}$$

$$b = -\frac{1}{2} \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \cdot (K(x_i, x_r) + K(x_i, x_s)) \tag{3.5.15}$$

[4] n corresponde al número de SVs

3.6 Mínimos Cuadrados para SVM

La técnica de los Mínimos Cuadrados — LS-SVM, del inglés *Least Squares Support Vector Machine* — fue propuesta y creada por [Suykens and Vandewalle, 1999]. Esta técnica es una reformulación de las SVM y soluciona el problema de regresión a través de un conjunto de ecuaciones lineales. De esta forma, este método puede tratar una cantidad más considerable de datos de entrenamiento y a la vez el costo de procesamiento de los datos de entrenamiento disminuye considerablemente. Una característica particular de SVM es el hecho de que la mayoría de los multiplicadores de Lagrange, asociados a los vectores de entrenamiento, son nulos. Esto no ocurre en LS-SVM, los valores se distribuyen, sin predominancia de valores nulos.

En el método LS-SVM se plantea el siguiente problema de minimización:

$$\begin{aligned} & min\, L_p(\omega, b, \xi) = \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{\ell}\xi_i^2 \\ & s.a. \\ & \qquad y_i = \langle \omega, \Phi(x_i)\rangle + b + \xi_i \qquad i = 1, \ldots, \ell \end{aligned} \tag{3.6.1}$$

Para resolver el problema de optimización (3.6.1) se define el siguiente funcional dual:

$$L_d(\omega, b, \xi, \alpha) = L_p(\omega, b, \xi) - \sum_{i=1}^{\ell}\alpha_i(\langle \omega, \Phi(x_i)\rangle + b + \xi_i - y_i), \tag{3.6.2}$$

sujeto a las siguientes condiciones

$$\begin{aligned} \frac{\partial L_d}{\partial \omega} = \omega - \sum_{i=1}^{\ell}\langle \alpha_i, \Phi(x_i)\rangle &= 0 \\ \frac{\partial L_d}{\partial b} = \sum_{i=1}^{\ell}\alpha_i &= 0 \\ \frac{\partial L_d}{\partial \xi_i} = \alpha_i - C\cdot\xi_i &= 0 \quad i = 1, \ldots, \ell \\ \frac{\partial L_d}{\partial \alpha_i} = \langle \omega, \Phi(x_i)\rangle + b + \xi_i - y_i &= 0 \quad i = 1, \ldots, \ell \end{aligned} \tag{3.6.3}$$

Por lo tanto, de la expresión anterior se deduce:

$$y_i = \sum_{i=1}^{\ell}\langle\langle \alpha_i, \Phi(x_i)\rangle ; \Phi(x_i)\rangle + b + \xi_i \tag{3.6.4}$$

Ahora si usamos una función Kernel $K(x_i, x_j)$ la expresión anterior quedaría:

$$y_i = \sum_{i=1}^{\ell} \alpha_i \cdot K(x_i, x_j) + b + \frac{\alpha_i}{C}. \tag{3.6.5}$$

Hasta aquí gracias a LS-SVM se ha logrado zafar de la desventaja de la resolución de un problema de optimización de programación cuadrática, a través de la resolución de un problema de ecuaciones lineales y así disminuir considerablemente el costo de procesamiento y aumentar el conjunto de entranamiento.

Capítulo 4

Optimización por Enjambre de Partículas

4.1 Introducción

El algoritmo de *Optimización por Enjambre de Partículas* — PSO, del inglés *Particle Swarm Optimization* — [Kennedy and Eberhart, 1995] es un algoritmo bioinspirado, que se basa en una analogía con el comportamiento social de ciertas especies. Se reconocen dos influencias básicas de inspiración en PSO:

1. El movimiento de bandadas de aves, en las que cada individuo se desplaza mediante reglas simples de ajuste de su velocidad en función de las observaciones que realiza sobre los individuos próximos en la bandada.
2. Un modelo social, en el que cada partícula representa una creencia, y las influencias entre individuos, la aproximación de unos individuos a otros por difusión de dichas creencias.

En líneas generales, el algoritmo busca encontrar el óptimo global de una función mediante el movimiento de un conjunto de *partículas* — enjambre — en un espacio definido por el número de parámetros de la función. Cada partícula es una posible solución, es decir, un conjunto de parámetros, que se evalúa calculando el valor de fitness que les correspondería. En dicho movimiento, las partículas utilizan información histórica — el resultado de su exploración pasada —, así como información sobre sus vecinos — otras partículas del enjambre —.

Por otro lado, para completar la especificación del algoritmo, hay que definir ciertos parámetros del algoritmo — criterio de parada, tipo de vecindario, valores de algunas constantes, forma de inicialización —. El enjambre comienza disperso por el espacio de búsqueda, pero con el tiempo las partículas convergen hacia una zona reducida del espacio en la que intensifican la búsqueda de la solución.

4.2 El paradigma de Inteligencia Colectiva

La *Inteligencia Colectiva* — SI, del inglés *Swarm Intelligence* — es un paradigma que agrupa técnicas de inteligencia artificial basadas en el estudio del comportamiento colectivo observado en la naturaleza. Este comportamiento es intrínseco a los orígenes de los individuos, los cuales tienen una fuerte tendencia a asociarse a otros y socializar [Kennedy, 2001]. De esta forma, reglas, conocimiento, experiencias y creencias pueden intercambiarse. La SI consta de una población de agentes simples interactuando localmente con los demás y con el ambiente que los rodea. En general no existe una estructura de control centralizada previendo cómo deberían actuar los agentes individuales. No obstante, las interacciones locales entre tales agentes a menudo convergen en un comportamiento global del cual todos se benefician. Ejemplos de esto son las colonias de hormigas, los cardúmenes, las bandadas y las manadas, en donde todos los agentes se mueven en conjunto. Se han efectuado diversos estudios para entender el comportamiento coordinado de los grupos, y éstos han sido utilizados en diferentes heurísticas de inteligencia colectiva. Una de ellas es la que se describe a continuación.

4.3 Terminología del PSO

Esta heurística se basa en el modelo psico-social de entidades colectivas. PSO posee influencia y aprendizaje social [Kennedy and Mendes, 2003], que se refleja en cada agente de la entidad. Los individuos — partículas — en la heurística emulan una característica simple: adaptan su comportamiento al de los individuos dentro de su propio vecindario o de la población completa — en inglés, *swarm* —.

Una partícula se mueve dentro del espacio de búsqueda siguiendo una trayectoria definida por su velocidad, y la memoria de dos buenos valores: el mejor encontrado por ella misma en su pasado y el mejor valor encontrado por alguna partícula de la población. Estos dos valores generan una "atracción" hacia los lugares más prometedores, y a esta atracción o fuerza se la denomina *presión social*.

4.3.1 Estructura de una Partícula

La estructura básica de una *partícula* es levemente más complicada que la de un individuo de un algoritmo genético. Consta de cinco componentes:

- Valor *objetivo* o aptitud *fitness*, representa la calidad de la solución representada por el vector $\overrightarrow{x}$, obtenido a través del cálculo de la función de evaluación correspondiente al problema específico.
- Un vector $\overrightarrow{v}$ que representa la *velocidad* de la partícula. Este vector, al igual que $\overrightarrow{x}$, se modifica en cada iteración del algoritmo reflejando así el cambio de dirección

que sufre la partícula dentro del espacio de búsqueda.

- *pbest* , mejor valor de fitness encontrado por la partícula hasta el momento.
- *gbest* , mejor valor de fitness encontrado por alguna partícula del enjambre — swarm —. Este valor está presente si se utiliza un modelo global, es decir, un modelo en el que los individuos son influenciados por el mejor de toda la población.
- *lbest* , mejor valor de fitness encontrado en el vecindario de una partícula. Este valor está presente si se utiliza un modelo local, es decir, un modelo en el que los individuos son influenciados por el mejor de un grupo pequeño de individuos — vecindario —. La determinación del vecindario depende de la forma en la que se efectúa el agrupamiento de individuos de la población.

4.3.2 Trayectoria de la Partícula

La trayectoria de la partícula dentro del espacio de búsqueda está definida con base en el vector de ubicación $\overrightarrow{x}$ y el de velocidades $\overrightarrow{v}$, los cuales son sumados para obtener la nueva dirección. Justamente estos cambios de trayectoria son los que determinan la característica principal del algoritmo PSO, ya que a través de los mismos las partículas son forzadas a buscar soluciones en las áreas más promisorias del espacio de búsqueda.

Cabe destacar que si los valores de $\overrightarrow{v}$ no se modificaran, la partícula sólo se movería con pasos uniformes en una única dirección. Más específicamente, en cada iteración del algoritmo, la dirección que tomaría la partícula es modificada considerando el valor de *pbest* — la atracción hacia la mejor posición alcanzada por la partícula — y el vector *gbest* — la atracción hacia la mejor posición alcanzada por alguna partícula del cúmulo o *lbest* si se consideran vecindarios —.

Además PSO introduce valores aleatorios de ajuste para las dos últimas atracciones. Estos valores aseguran que el tamaño del paso que realizan las partículas dentro del espacio sea variable, asegurando que las mismas no "caigan en una rutina", es decir, que no se muevan siempre con el mismo paso. La Figura 4.1 ilustra la obtención de la nueva posición — *x*'— de la partícula *x* como consecuencia de la suma de la velocidad y las memorias de los mejores.

La influencia relativa de la memoria que lleva asociada cada partícula — pbest — es denominada *influencia cognitiva*, mientras que la memoria del mejor valor encontrado por alguna partícula de la población — gbest o lbest — es denominada *influencia social*. Ambas influencias determinan la velocidad de aprendizaje del cúmulo y esto permite determinar con qué rapidez el cúmulo convergerá a la solución.

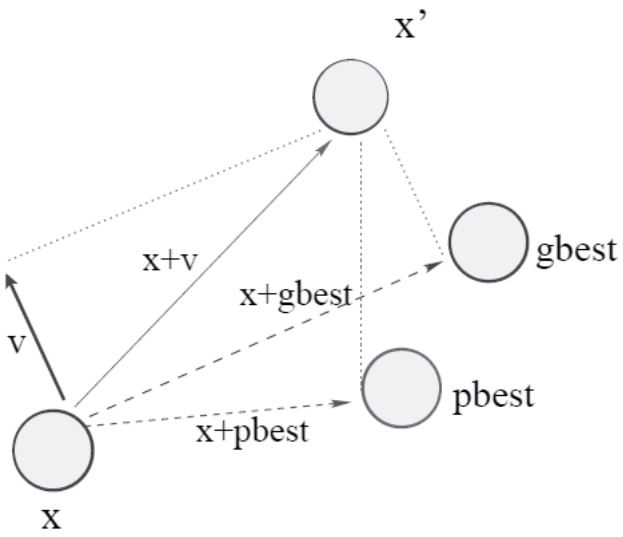


Figura 4.1: *Trayectoria de la partícula x*

Matemáticamente, el proceso de aprendizaje puede describirse utilizando una fórmula para la actualización de la velocidad de cada partícula y otra para la actualización de su posición dentro del espacio de búsqueda.

4.4 Algoritmo

Considerando un problema de D-dimensiones, para cada partícula i de d dimensiones con $d \in [1, D]$, el aprendizaje está determinado por las siguientes ecuaciones:

$$v_{id} = v_{id} + \rho_1 \cdot r_1 \cdot (pbest_{id} - x_{id}) + \rho_2 \cdot r_2 \cdot (pbest_d - x_{id}) \tag{4.4.1}$$

$$x_{id} = x_{id} + v_{id} \tag{4.4.2}$$

donde v_{id} es el valor de la velocidad de la partícula i en la dimensión d, ρ_1 es el factor de aprendizaje cognitivo, ρ_2 es el factor de aprendizaje social, r_1 y r_2 son valores aleatorios uniformemente distribuidos en el rango $[0, 1]$, x_{id} es la posición corriente de la partícula i en la dimensión d, $pbest_{id}$ es el valor en la dimensión d de la partícula con el mejor valor objetivo encontrado por la partícula i, y $gbest_d$ es el valor en la dimensión d del individuo del enjambre — swarm — que encontró el mejor valor objetivo. Como puede observarse en la fórmula, un incremento de ρ_2 sobre ρ_1 aumenta la influencia del valor *gbest*, lo cual resulta en una mayor *exploración* del espacio de soluciones; decrementando el valor de ρ_2 sobre ρ_1 causa que la partícula se mueva en dirección más cercana a *pbest*, lo cual resulta en una mayor *explotación* del espacio. Cuando se habla de exploración se considera la habilidad del algoritmo para explorar las diferentes regiones del espacio de

búsqueda a fin de encontrar buenas soluciones. Mientras que la explotación es la habilidad que el algoritmo posee de concentrarse en una porción del espacio que sea promisoria con el fin de refinar soluciones buenas, y encontrar posiblemente el óptimo.

Además de estas formulas, un algoritmo PSO cuenta con los siguientes parámetros:

- N: número de partículas involucradas en la resolución del problema.
- D: número de variables del problema, equivalente al número de dimensiones de una partícula.
- $Xmax$ y $Xmin$: parámetros que limitan el área de búsqueda.
- $Vmin$ y $Vmax$: parámetros opcionales que limitan la velocidad de las partículas. Son opcionales porque si bien es necesaria una restricción de los valores que la velocidad puede alcanzar, existen otros métodos como el factor de constricción que cumplen con la finalidad de limitar dichos valores.
- *Condición de Parada* del algoritmo: puede usarse un criterio que establezca un nivel de error aceptable entre el ideal y el óptimo obtenido, una cantidad máxima de iteraciones — ciclos de vuelo — o alguna otra que se prefiera.

Observando el Algoritmo 4.1, la fase de Inicialización del enjambre asigna a cada dimensión de cada partícula de la población un valor aleatorio en el rango establecido por [$Xmin$,$Xmax$]. Este rango depende exclusivamente del problema que se pretende resolver con la heurística. Lo mismo sucede con los vectores de velocidad — uno por cada partícula—, los que son inicializados en un rango [$Vmin$,$Vmax$]. Los valores objetivo son calculados y almacenados. A cada partícula ***pbest*** se le copia el valor de cada dimensión de su individuo correspondiente, al igual que la partícula ***gbest*** a la que se le asigna el individuo con mejor valor objetivo. La segunda y más importante es la fase de búsqueda, en la cual el enjambre recorre el espacio para hallar la mejor solución posible. En un proceso repetitivo — hasta que se cumpla una condición de parada —, se actualizan las fórmulas de aprendizaje — velocidad y posición — con base en los valores actuales de ***pbest*** y ***gbest***. Los valores de fitness son recalculados y utilizados para reemplazar las partículas ***pbest*** y ***gbest***, si es que en la presente iteración se obtuvieron mejores valores objetivo.

Algoritmo 4.1 Algoritmo PSO Básico

(* *Fase de Inicialización del enjambre* *)

PARA i **HASTA** N **HACER**
  PARA d **HASTA** D **HACER**
    $x_{id} := Random(Xmin, Xmax)$
    $pbest_{id} := x_{id}$
    $v_{id} := Random(Vmin, Vmax)$
  FIN

  $fitness_{x_i} := Fitness(x_i)$
  $fitness_{pbest_i} := fitness_{x_i}$
  SI $fitness_{x_i}$ mejor $fitness_{gbest}$ **ENTONCES**
    $gbest := x_i$
  FIN
FIN

(* *Fase de Búsqueda* *)

MIENTRAS no se cumpla condición de parada **HACER**
  PARA i **HASTA** N **HACER**
    PARA d **HASTA** D **HACER**
      $x_{id} := Posicion()$
      $v_{id} := Velocidad()$
    FIN

    $fitness_{x_i} := Fitness(x_i)$

  SI $fitness_{x_i}$ mejor $fitness_{gbest}$ **ENTONCES**
    $gbest := x_i$
  FIN

  SI $fitness_{x_i}$ mejor $pbest_i$ **ENTONCES**
    $pbest_i := x_i$
    $fitness_{pbest_i} := Fitness(pbest_i)$
  FIN
FIN

4.5 Vecindarios en PSO

Una de las principales características que posee PSO es la interacción social que se observa entre los individuos. Por eso es importante el estudio de las posibles estructuras sociales que pueden incluirse en PSO. Todos los individuos pertenecientes a la misma

estructura social comparten información, aprenden y siguen a los mejores dentro de su propia estructura.

La comunicación entre los individuos de un grupo o población así como su desempeño, están íntimamente ligados a la estructura social que posee el grupo. Así también, en PSO, dichas interacciones se observan particularmente entre los vecinos inmediatos adyacentes, y son estas interacciones las que provocan que el algoritmo funcione, ya que cada partícula por sí misma no podría evolucionar demasiado.

La estructura más simple que se puede plantear es aquella donde todos los individuos son influenciados exactamente de la misma forma, por la mejor solución encontrada por algún miembro de la población. Este tipo de modelo recibe el nombre de *gbest* — mejor global —, y es equivalente a una estructura donde todos los individuos están conectados entre sí. Otra estructura que surge naturalmente recibe el nombre de *lbest* — mejor local —.

La elección de la estructura social es uno de los principales inconvenientes que presenta la implementación de PSO, debido a la importancia y fuerte influencia que ésta ejerce en el desempeño del algoritmo. El modelo *gbest* tiende a converger prematuramente, debido a la pérdida de la diversidad. El modelo *lbest* es más lento de converger, pero conserva la diversidad [Kennedy, 2001]. Los resultados obtenidos con algoritmos que incluyen vecindarios suelen ser, generalmente, mejores cuando son comparados con los obtenidos con modelos *gbest*.

Varias estructuras *lbest* pueden ser utilizadas, algunas de estas son [Kennedy, 1999]:

- *Circular o Anular*: cada individuo está conectado con sus k vecinos inmediatos, intentando imitar al mejor de ellos. En la Figura 4.2a puede observarse un ejemplo con $k = 2$. En este tipo de estructura, cada vecindario se encuentra solapado para facilitar el intercambio de información y, de esta manera, converger a una única solución al final del proceso de búsqueda. La información entre vecindarios fluye lentamente, lo cual provoca que la convergencia sea más lenta pero al mismo tiempo más segura. En funciones multimodales — aquéllas que poseen varios óptimos — este tipo de estructura suele obtener un buen desempeño con respecto a otras estructuras. Muchas variantes de este tipo de estructura pueden ser creadas, en donde la conexión de las partículas puede ser elegida aleatoriamente o siguiendo algún criterio, dependiendo del problema. La Figura 4.2b muestra un ejemplo.

- *Rueda*: un único individuo — el central — está conectado a todos los demás, y todos éstos están conectados solamente al individuo central. La Figura 4.2c ilustra el concepto. En esta estructura una única partícula sirve de punto focal, y toda la información entre los individuos es canalizada por ese punto. La partícula focal compara el desempeño de todas las partículas en el vecindario para ajustar las posiciones a la mejor. Tan pronto como la mejor posición sea detectada, la información es enviada a todas las partículas para que efectúen el cambio. La estructura de rueda propaga las buenas soluciones en forma demasiado lenta lo

cual puede no ser del todo bueno, porque el proceso de búsqueda se tornaría demasiado lento. Algunas variantes pueden ser introducidas, como desconectar algunas partículas de la partícula focal y conectarlas a otras partículas en el vecindario — Figura 4.2d —.

- *Arcos Aleatorios*: para N partículas, N conexiones simétricas son asignadas entre pares de individuos. Como su nombre lo indica, las conexiones son asignadas de forma aleatoria entre las N partículas. La Figura 4.2e muestra esta estructura para un tamaño de población de 12 individuos.

- *Pirámide*: la idea de esta topología es la creación de una estructura tridimensional que comunique a las partículas. Se asemeja a un modelo de alambre — wire-frame — tridimensional, como se observa en la Figura 4.2f.

- *Toroidal*: Esta topología puede utilizarse como estructura de vecindario [Gardner, 1975]. Es una superficie cerrada con base en la topología de malla — Figura 4.2g — pero a diferencia de esta última donde cada partícula posee dos vecinos, en la toroidal posee 4 vecinos directos — Figura 4.2h —.

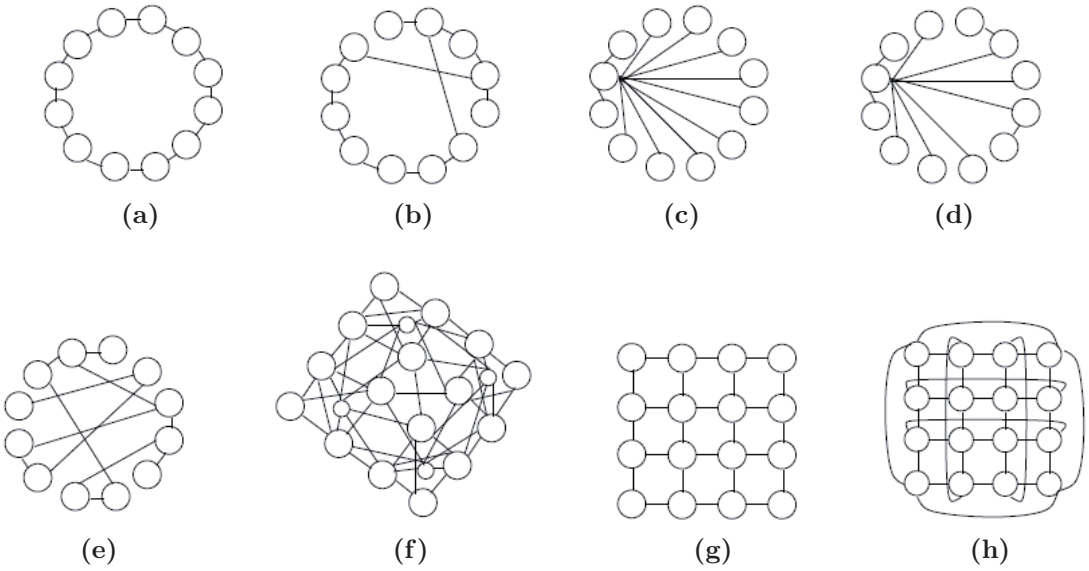


Figura 4.2: *Topología de Vecindarios de Partículas.*

Cada una de estas topologías, a su vez, puede variar algunas de sus conexiones, para constituir subvecindarios. Lo importante es lograr una topología que mejore el desempeño del algoritmo. Actualmente, no existe una topología que sea buena para todo tipo de problema. Generalmente, las estructuras completamente conectadas son mejores para problemas unimodales — un único óptimo —, mientras que las menos conectadas funcionan mejor para problemas multimodales, dependiendo del grado de interconexión entre las partículas [Kennedy, 1999] y [Peer and Engelbrecht, 2003].

4.6 Variantes del Algoritmo PSO

Durante la primera década de la utilización del algoritmo PSO, aparecieron numerosos trabajos que, partiendo del enjambre original, proponían modificaciones con el fin de mejorar sus características, normalmente de forma empírica. Asimismo aparecieron algunos trabajos que realizaron análisis teóricos de las ecuaciones del enjambre, con el fin de comprender mejor el mecanismo de su funcionamiento: la estabilidad, convergencia, estudio de trayectorias e influencia de los parámetros.

Algunos de los problemas de las primeras versiones de PSO se debían a su sensibilidad a la elección de los parámetros. Los valores escogidos no sólo condicionaban la posibilidad de encontrar el óptimo global de la función de fitness; se descubrió además el fenómeno de explosión del enjambre. Éste consiste en que, con ciertos valores de los parámetros, las velocidades de las partículas — v_i — tienden a crecer hasta valores elevados, dispersando el enjambre de forma que la distancia entre partículas era cada vez mayor.

Con el fin de estudiar la convergencia del algoritmo, y limitar la explosión del enjambre, se estudiaron varios mecanismos:

4.6.1 Control de la Velocidad

Consiste simplemente en fijar un límite superior al valor absoluto de las componentes de velocidad. Es decir, se fijaba un parámetro $Vmax$ y se imponía la condición de limitación siguiente:

$$v_i^{t+1} = min(v_i^{t+1}, Vmax). \tag{4.6.1}$$

Este mecanismo tiene el problema de que el parámetro $Vmax$ depende del tamaño del espacio de búsqueda $Xmax$, ya que en espacios de búsqueda grandes, velocidades pequeñas supondrían que el enjambre no recorrería todo el espacio o necesitaría un número de iteraciones elevado para hacerlo. En los trabajos iniciales se solía proponer el valor $Vmax = Xmax$.

4.6.2 Factor de Inercia (IWPSO)

Se introdujo un parámetro w — ver Ecuación (4.6.2)— llamado *factor de inercia* [Shi and Eberhart, 1998]. Se argumenta que el término de inercia de la ecuación es importante para que el enjambre pueda realizar búsquedas globales, más allá de las fronteras del espacio definido por las posiciones de las partículas iniciales. La ecuación de velocidad (4.4.1) se reemplaza por:

$$v_{id} = w \cdot v_{id} + \rho_1 \cdot r_1 \cdot (pbest_{id} - x_{id}) + \rho_2 \cdot r_2 \cdot (pbest_d - x_{id}). \tag{4.6.2}$$

Los autores proponen incluir un parámetro explícito con el objetivo de permitir calibrar el equilibrio entre exploración y explotación, al añadir la posibilidad de variar el valor de dicho parámetro en función de la iteración o estado del algoritmo. En su análisis concluyen que, cuando el valor de w es pequeño, el enjambre tiende a comportarse como un algoritmo que realiza búsquedas locales, y depende con más fuerza de la inicialización. Cuando w crece, pasa a realizar una búsqueda más global, pero encuentra el óptimo con más dificultad y mayor número de iteraciones. Mediante el estudio experimental realizado, se propone un valor de compromiso $0{,}9 < w < 1{,}2$.

En el trabajo, se propone también el uso de un valor de w decreciente con el tiempo. Con posterioridad, esta propuesta se analizó de forma detallada en [Shi and Eberhart, 1999]. Aunque en este estudio el factor de inercia decreciente parece mejorar el comportamiento del enjambre, en [Zheng and Qian, 2003] se muestra un estudio en el que PSO tiene mejores resultados cuando se utiliza un valor de w que crece con el tiempo en lugar de decrecer — ver ecuación (4.6.3) —. Por lo tanto, el comportamiento resulta ser dependiente de los problemas considerados.

$$w = w_{max} - \frac{(w_{max} - w_{min})}{iter_{max}} k. \tag{4.6.3}$$

donde w_{max} corresponde al valor máximo del factor de inercia, w_{min} es el valor mínimo que puede alcanzar el factor de inercia, $iter_{max}$ representa el numero máximo de iteraciones del algoritmo y finalmente, k hace referencia a la iteración actual del algoritmo.

4.6.3 Landscape Adaptive (LAPSO)

Esta variante nace de la hipótesis de que la distribución de todo el grupo de partículas puede entregar información adicional importante al problema [Yisu et al., 2008]. Durante el proceso de búsqueda esta distribución está sujeta a un constante cambio. Para capturar estas variaciones es que se define el vector de distribución $\overrightarrow{D}$, para describir el espacio de las partículas.

$$\overrightarrow{D_j} = \frac{max_{i=1}^{d}(x_{ij}) - min_{i=1}^{d}(x_{ij})}{abs(max_{i=1}^{d}(x_{ij}) + abs(min_{i=1}^{d}(x_{ij})))},\ j = 1,2,3,\ldots,m \tag{4.6.4}$$

El valor del vector $\overrightarrow{D}$ estará entre 0 y 1 entregando información en la dirección de las componentes. La ecuación de velocidad de la partícula quedaría expresada de la siguiente manera:

$$v_{id} = \overrightarrow{D} \cdot (v_{id} + \rho_1 \cdot r_1 \cdot (pbest_{id} - x_{id}) + \rho_2 \cdot r_2 \cdot (pbest_d - x_{id})), \tag{4.6.5}$$

Esta variante puede conllevar a una prematura convergencia del algoritmo y posiblemente a óptimos no deseados, para evitar este problema es que se hace una redefinición del valor del vector $\overrightarrow{D}$:

$$\overrightarrow{D} = \begin{cases} D_{max} & D_j < D_{min} \\ D_{min} & D_j > D_{max} \\ D_j & e.o.c. \end{cases} \tag{4.6.6}$$

Donde D_{min} y D_{max} son parámetros del algoritmo. Los valores recomendados por [Yisu et al., 2008], son 0.4 y 0.95 respectivamente.

4.6.4 Adaptive Mutation (AMPSO)

Esta modificación plantea el concepto de mutación en la mejor posición de las partículas a lo largo de la búsqueda [Pant et al., 2008b]. Las partículas van mutando por cada iteración de acuerdo a la siguiente regla:

$$x_{id} = x_{id} + \sigma'_{id} \cdot \beta_i \tag{4.6.7}$$

donde:

$$\sigma'_{id} = \sigma_{id} \cdot exp(\lambda \cdot R_1 + \lambda' \cdot R_2) \tag{4.6.8}$$

R_1 y R_2 denotan un número aleatorio de distribución normal $N(0,1)$, mientras que β_i corresponde a un valor aleatorio de distribución beta. Además se tiene que λ y λ' corresponden a:

$$\lambda = \frac{1}{\sqrt{2n}},\ \lambda' = \frac{1}{\sqrt{2\sqrt{n}}} \tag{4.6.9}$$

4.6.5 Parámetros de Constricción

Se propone una alternativa a la inclusión del factor de inercia [Clerc, 1999] , que se estudia con más detalle en [Clerc and Kennedy, 2002]. En este trabajo se realizó un estudio teórico de la dinámica del enjambre y se observó que se podía garantizar la convergencia del algoritmo en ciertas condiciones mediante la inclusión de parámetros adicionales llamados *parámetros de constricción* — c —. Estos resultados se obtienen mediante el análisis de versiones simplificadas del enjambre, y específicamente versiones en las que se eliminan los factores aleatorios. Con posterioridad se confirman estas propiedades empíricamente usando la versión completa del enjambre. Los parámetros de constricción son una alternativa al mecanismo de inercia y de limitación de velocidad. La ecuación de velocidad (4.4.1) se reemplaza por la (4.6.10).

$$v_{id} = c \cdot (v_{id} + \rho_1 \cdot r_1 \cdot (pbest_{id} - x_{id}) + \rho_2 \cdot r_2 \cdot (pbest_d - x_{id})), \tag{4.6.10}$$

donde el factor c se define como:

$$c = \frac{2k}{\sqrt{\left|2 - \phi - \sqrt{\phi(\phi - 4)}\right|}}, \tag{4.6.11}$$

siendo:

$$\begin{gathered} \phi = \phi_1 + \phi_2. \\ \phi > 4. \\ \phi_1 = \rho_1 \cdot r_1. \\ \phi_2 = \rho_2 \cdot r_2. \\ k \in [0, 1]. \end{gathered} \tag{4.6.12}$$

El parámetro k en la ecuación (4.6.11) controla la exploración y explotación del cúmulo. Para valores de k cercanos a 0, se obtiene una rápida convergencia con explotación local. Para valores de k cercanos a 1, la convergencia se produce más lentamente y se experimenta un grado mayor de exploración. Aunque el valor de k es constante, algunas variaciones comienzan con valores altos y, a medida que transcurren los ciclos, se va decrementando a fin de aumentar la explotación.

4.6.6 Quantum PSO (QPSO)

Según el modelo clásico de PSO, el estado de las partículas está dado por el vector de posición x_i y el vector de velocidad v_i, determinando así la trayectoria de las partículas, las que se mueven siguiendo los principios básicos de la mecánica Newtoniana. Sin embargo, si consideramos la mecánica cuántica, el comportamiento dinámico de la partícula difiere ampliamente del comportamiento tradicional de PSO, donde los valores exactos de velocidades y posiciones no pueden ser determinados simultáneamente.

Por lo tanto, si las partículas pertenecientes al enjambre tienen un comportamiento cuántico, el rendimiento del algoritmo PSO estará muy lejos del PSO clásico.

En el modelo cuántico de PSO, el estado de cada partícula está representado por la función de onda $\psi(x,t)$, en vez de la posición y la velocidad. De acuerdo al significado estadístico de la función de onda, la probabilidad de que una partícula aparezca en una

determinada posición se puede obtener de la función de probabilidad de la densidad $|\psi(x,t)|^2$.

En QPSO, las partículas se mueven de acuerdo a las siguientes funciones [Pant et al., 2008a]:

$$x(t+1) = p + \beta * |mbest - x(t)| * ln(\frac{1}{n});\ Si\ k \geq 0{,}5 \tag{4.6.13}$$

$$x(t+1) = p - \beta * |mbest - x(t)| * ln(\frac{1}{n});\ Si\ k \geq 0{,}5 \tag{4.6.14}$$

donde:

$$p = \frac{(c_1 p_{ij} + c_2 p_{gj})}{c_1 + c_2} \tag{4.6.15}$$

$$mbest = \frac{1}{M}\sum_{i=1}^{M} p_i = (\frac{1}{M}\sum_{i=1}^{M} p_{i1}, \frac{1}{M}\sum_{i=1}^{M} p_{i2}, ..., \frac{1}{M}\sum_{i=1}^{M} p_{iM}) \tag{4.6.16}$$

4.6.7 PSO con Adaptación Dinámica (DAPSO)

En el PSO convencional, las partículas siguen dos valores: el valor actual de la mejor solución global y la mejor solución encontrada hasta el momento por la partícula. Es así, que las velocidades de las partículas convergen rápidamente a cero, haciendo que las partículas se encuentren atrapadas en óptimos locales. Este fenómeno se denomina "similitud" de enjambre de partículas y provoca que se disminuya el área de búsqueda de partículas. En [Yang et al., 2007] proponen que para ampliar la zona de búsqueda se puede utilizar las siguientes alternativas: por un lado aumentar el número de partículas o bien, debilitar la capacidad de las partículas para seguir al valor de la mejor solución del enjambre. Sin embargo, la primera opción implica una mayor complejidad computacional, mientras que la última, provoca una convergencia lenta del algoritmo.

Los autores incorporan a la fórmula convencional de velocidad un factor de inercia w, el cual varía en cada iteración. Sin embargo, si este factor decrece linealmente no se refleja el proceso actual de búsqueda. Es por ello, que en [Yang et al., 2007] proponen que el factor de inercia sea afectado por el estado evolutivo del algoritmo y que sea determinado por el factor de velocidad evolutivo de cada partícula y el grado de agregación del enjambre.

De esta manera, los autores establecen que el factor de velocidad evolutivo esta dado por la siguiente ecuación:

$$h_i^t = \left| \frac{min(F(pbest_i^{t-1}), F(pbest_i^t))}{max(F(pbest_i^{t-1}), F(pbest_i^t))} \right| \tag{4.6.17}$$

donde $F(pbest_i^t)$ es el valor fitness de $pbest_i^t$. Bajo este supuesto, se obtiene que $0 < h \leq 1$. Este parámetro toma en cuenta la historia recorrida por cada partícula y refleja la evolución de la velocidad de cada partícula, es decir, a menor valor de h, más rápida es la velocidad.

En cambio, el factor el grado de agregación viene dado por la siguiente ecuación:

$$s = \left| \frac{min(F_{tbest}, \overline{F}_t)}{max(F_{tbest}, \overline{F}_t)} \right| \tag{4.6.18}$$

donde $\overline{F}_t$ corresponde a la media del valor fitness de todas las partículas del enjambre en la t-ésima iteración.

De esta manera, w puede ser determinado como una función de los parámetros h y s: $w_i^t = f(h_i^t, s)$.

El propósito de la variación de w, es dar al algoritmo una mejor capacidad de búsqueda y que pueda salir de óptimos locales. Mientras tanto, con el fin de evitar el efecto de similitud del enjambre, la capacidad de salir de óptimos locales debiese ser mejorada, es decir, cuando el grado de agregación se hace más grande, el peso de inercia debería incrementar proporcionalmente. Es por ello, que los autores proponen que el factor de inercia se obtenga a partir de la siguiente ecuación:

$$w_i^t = w_{ini} - \alpha(1 - h_i^t) + \beta s \tag{4.6.19}$$

Donde w_{ini} es el valor inicial de w y es inicializado en 1. Los valores de α y β se encuentran habitualmente dentro del rango [0,1].

4.6.8 Influencia de los Factores Cognitivo y Social

Dependiendo de la importancia de los factores cognitivo y social, valores para φ_1 y φ_2 respectivamente, se pueden identificar 3 tipos de algoritmos:

- **Modelo completo**: Para este caso, tanto el componente cognitivo como el social intervienen en el movimiento de las partículas, es decir, $\varphi_1, \varphi_2 > 0$.
- **Modelo sólo cognitivo**: En el algoritmo PSO, únicamente el componente cognitivo participa en el movimiento de las partículas, por lo cual, los valores para φ_1 deben ser mayores a 0, mientras que para φ_2 deben ser iguales a cero.
- **Modelo sólo social:** De forma contraria al modelo anterior, el movimiento de las partículas se ve influenciado sólo por el componente social, de esta manera, $\varphi_1 = 0$ y $\varphi_2 > 0$.

4.6.9 Según el Tipo de Vecindario

Desde el punto de vista del vecindario, considerando la cantidad y posición de las partículas que intervienen en el cálculo de la distancia en la componente social, se pueden determinar dos tipos de algoritmos:

- **PSO local**: En esta versión, se calcula la distancia entre la posición actual de la partícula y la posición de la mejor partícula encontrada dentro de su entorno local, el cual consiste en las partículas inmediatamente cercanas en la topología del enjambre.
- **PSO global:** En este tipo de algoritmo PSO, la distancia en la componente social viene dada por la diferencia entre la posición de la partícula actual y la posición de la mejor partícula encontrada en el enjambre completo.

La convergencia del algoritmo es más rápida a través de la versión Global de PSO, debido a que la visibilidad de cada partícula es mejor y se acercan más a la mejor posición dentro del enjambre, favoreciendo la intensificación, por está razón, se cae más fácilmente en óptimos locales, impidiendo que se exploren otras regiones del espacio de búsqueda. De forma contraria, la convergencia en más lenta en el PSO Local, favoreciendo en este caso la diversificación, pero a diferencia de PSO Global no se cae de manera sencilla en óptimos locales [Sedighizadeh and Masehian, 2009].

Capítulo 5

Modelo LS-SVM + PSO

5.1 Introducción

En este capítulo, se presenta la estructura general de los modelos desarrollados para pronosticar los niveles de desembarques mensuales de anchovetas de la zona norte de Chile. Para lograr dicho objetivo, es que se hará uso de las técnicas expuestas en los capítulos anteriores.

El eje central de los modelos son las *Máquinas de Soporte Vectorial para Regresión* — SVM — a través de su reformulación de Mínimos Cuadrados en la cual se hace uso de un conjunto de ecuaciones lineales para definir el problema — presentado en 3.6—, esto traerá como consecuencia una alta disminución en el procesamiento de los datos. La tarea más compleja a la hora de utilizar SVMs, corresponde a la búsqueda de los parámetros óptimos de solución, dicha configuración será determinante para conseguir una alta bondad en los modelos.

Para darle solución al caso expuesto en el párrafo anterior, es que se recurrirá al uso de la técnica de optimización combinatoria llamada *Particle Swarm Optimization* — PSO — y algunas de sus variantes — definida en el capítulo 4—, con el afán de encontrar dentro del espacio de búsqueda, los valores óptimos de los parámetros de la SVM. La metodología de discriminación, selección y evaluación de la calidad de estos parámetros se hará a través de la función *fitness* del algoritmo PSO, dicha función será representada a través de dos métricas, el *Error Cuadrático Medio* y la *Validación Cruzada — Cross Validation* en inglés —.

Ya teniendo a disposición los valores de los parámetros, se procederá a la evaluación de los diversos modelos obtenidos a través de métricas pertinentes, con el afán de seleccionar el mejor de ellos. Todo este proceso será abordado a mayor cabalidad a lo largo del presente capítulo, en donde se describirán los datos, la estructura interna del modelo, el proceso de obtención de los parámetros iniciales del algoritmo PSO y las diversas métricas y funciones fitness utilizadas para lograr el fin deseado.

5.2 Software y Hardware utilizado

La implementación se ha realizado a través del software matemático *Matlab* en su versión R2011a, el cual ofrece un entorno de desarrollo integrado — *IDE* — y su propio lenguaje de programación denominado M. Por medio de esta plataforma, se pudo hacer uso del *Toolbox LS-SVM 1.7*, el cual provee una serie de funcionalidades LS-SVM adaptables al problema en cuestión [De Brabanter et al., 2010]. Es a través de este entorno de desarrollo, que se han implementado tanto el algoritmo PSO junto con sus variantes, así como también el diseño de las interfaces y el análisis de los resultados.

Las características del sistema que se dispuso para la realización de las pruebas fueron las siguientes:

- Procesador Intel(R) Pentium(R) Dual Core T2390 1.86 GHz.
- Memoria RAM 3.0 Gb SoDimm DDR2 667 MHz.
- Sistema Operativo GNU/Linux Debian 6.0 64 bits.

5.3 Funciones Fitness

Se han utilizado las siguientes funciones fitness para medir el grado de adecuación de las partículas al problema:

5.3.1 Error Cuadrático Medio

El *Error Cuadrático Medio* — MSE —, corresponde a una de las técnicas más utilizadas para medir la calidad y posterior selección de un modelo que busque explicar una serie de datos en particular. Es por esto, que el MSE será utilizado para determinar la mejor configuración de los parámetros internos de la SVM. Para lograr encontrar los valores adecuados de dichos parámetros es que se hará uso de la ecuación (5.4.2), la cual busca medir el error global del pronóstico sujeto a una configuración parámetros particular.

5.3.2 Validación Cruzada

La *Validación Cruzada — Cross Validation* en inglés —, es la práctica estadística de partir una muestra de datos en subconjuntos de tal modo que el análisis es inicialmente realizado en uno de ellos, mientras los otros subconjuntos son retenidos para su uso posterior en la confirmación y validación del análisis inicial. Es una técnica muy utilizada en Inteligencia Artificial para validar los modelos generados a partir de un conjunto de datos o muestra.

La validación simple consiste en dividir en dos conjuntos complementarios los datos de la muestra, usar uno de ellos para construir el modelo — datos de entrenamiento — y usar el otro para medir el ratio de error del modelo construido — datos de validación —.

La validación cruzada más utilizada es la denominada *K-pliegues — K-fold Cross Validation* —- la cual aplica k-veces la validación simple, dividiendo en k conjuntos la muestra. En cada iteración se construirá y evaluará un modelo, usando uno de los conjuntos como datos de prueba y el resto como datos de entrenamiento. Al final obteniendo la media aritmética de los ratios de error obtenidos conseguiremos el ratio de error para la muestra final. La elección del valor k dependerá del tamaño y características de la muestra, pero un valor muy utilizado es 10-fold, el cual también es utilizado en este modelo.

5.4 Métricas de Evaluación

Una técnica de evaluación de modelos se define como un amplio conjunto de contrastes a los cuales un modelo puede y debe someterse en muy diferentes etapas durante el proceso de construcción y subsiguiente empleo. Existen una gran cantidad de índices que buscan medir la bondad de un modelo, la elección se hará en base a los objetivos que persigue el investigador. Por lo general, para medir la eficacia de los modelos se realiza una división de la muestra en dos grupos, la primera parte de la muestra es con la cual construye el modelo y con la segunda se evalúa la eficacia de este mismo — análisis fuera de la muestra —.

A continuación se presentan algunos métodos o índices utilizados para medir la capacidad de un modelo de regresión para una serie temporal [Hyndman and Koehler, 2006], donde:

- y_t representa el valor observado en un periodo t.
- $\hat{y}_t$ denota el valor pronosticado por el modelo correspondiente al periodo t.
- $\bar{y}$ es la media de los datos observados.
- $e = y_t - \hat{y}_t$ corresponde al error residual del pronóstico.
- n corresponde al número de periodos considerados en el estudio.

5.4.1 Coeficiente de Determinación

El *Coeficiente de Determinación* — en corto R^2 — se interpreta como la proporción de la variación de la variable endógena — real o de entrada — que queda explicada por la regresión. En otras palabras mide la dependencia que existe entre los datos reales y los pronosticados. Un valor muy cercano a 0 muestra independencia y dependencia en el caso el valor se acerque a 1. Se calcula de la siguiente manera:

$$R^2 = 1 - \frac{\sum_{t=1}^{n} (y_t - \hat{y}_t)^2}{\sum_{t=1}^{n} (y_t - \bar{y})}, \tag{5.4.1}$$

donde $\sum_{t=1}^{n} (y_t - \hat{y}_t)^2$ corresponde a la varianza de los errores y $\sum_{t=1}^{n} (y_t - \bar{y})$ a la varianza de la variable real.

5.4.2 Error Cuadrático Medio

El *Error Cuadrático Medio* — MSE, del inglés *Mean Square Error* — consiste en la suma de las diferencias al cuadrado entre lo real y lo proyectado por el modelo. Su ecuación es la siguiente:

$$MSE = \frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \tag{5.4.2}$$

5.4.3 Raíz del Error Cuadrático Medio

El *Error Cuadrático Medio* — RMSE, del inglés *Root Mean Square Error* — corresponde a la raíz cuadrada del MSE y es una estimación de la desviación estándar del error de predicción. Se calcula a través de la siguiente ecuación:

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2}. \tag{5.4.3}$$

5.4.4 Porcentaje de Error Medio Absoluto

El *Porcentaje de Error Medio Absoluto* — MAPE, del inglés *Mean Absolute Percentage Error* — proporciona una indicación de que tan grandes son los errores de pronóstico comparados con los valores reales de la serie. También, correspondiente a la operación interna de la sumatoria se encuentra el Porcentaje de Error Absoluto — APE en corto —. Estas medida tienen la desventaja de llegar a valores infinitos o indefinidos, si $y_t = 0$ para cualquier periodo t, y que tenga una distribución muy desigual cuando cualquier valor de y_t es cercano a cero [Hyndman and Koehler, 2006].

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{(y_t - \hat{y}_t)}{y_t} \right| \times 100 \, , y_t \neq 0. \tag{5.4.4}$$

5.4.5 Escala de Error Medio Absoluto

La *Escala de Error Medio Absoluto* — MASE, del inglés *Mean Absolute Scale Error* — es una aplicación general para obtener la exactitud del pronóstico, sin presentar los problemas vistos en el MAPE. Esta métrica se adapta bien a la demanda de series intermitentes porque nunca entrega valores de infinitos o indefinidos. Su ecuación es la siguiente:

$$MASE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{(y_t - \hat{y}_t)}{\frac{1}{n-1}\sum_{i=2}^{n} y_i - y_{i-1}}\right|. \tag{5.4.5}$$

donde $\frac{1}{n-1}\sum_{i=2}^{n} y_i - y_{i-1}$ corresponde al error de pronóstico promedio de un punto a otro, también denominada método de pronóstico informal — *naive forecast method* en inglés —. Este método utiliza el valor real del período anterior y_{i-1} como el pronóstico.

5.5 Descripción del Modelo General

El modelo general, consiste en una *Máquina de Soporte Vectorial de Mínimos Cuadrados* — LS-SVM —, cuyos parámetros internos serán obtenidos a través del método de optimización computacional *Particle Swarm Optimization* — PSO —. La idea central es que cada partícula perteneciente al enjambre simbolice una posible solución perteneciente al espacio de búsqueda, de esta manera se logrará a través de un proceso iterativo obtener aquella solución que bajo los criterios de evaluación, represente de mejor manera el comportamiento de los datos.

A continuación se presentará el conjunto de pasos que permitirán llevar a cabo el procedimiento anteriormente descrito:

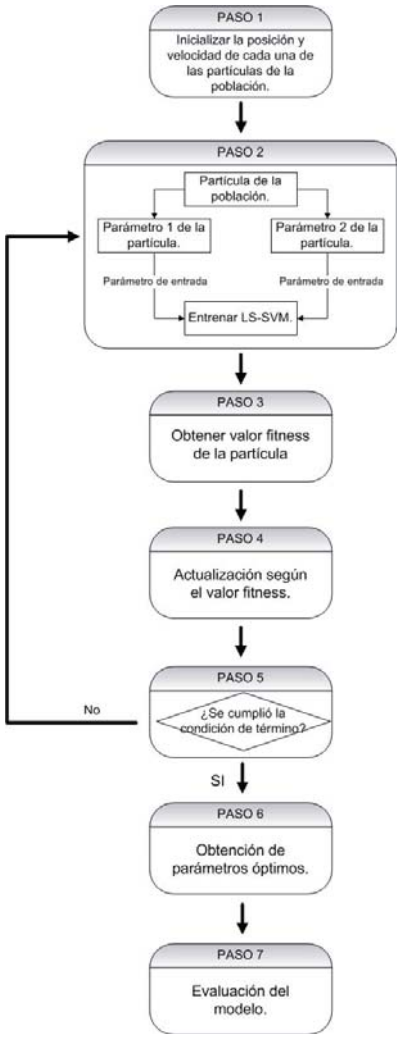


Figura 5.1: *Modelo General LS-SVM-PSO.*

- **Paso 1: Inicialización.** El algoritmo PSO es inicializado con una población de partículas, donde a cada una de ellas se les asigna posiciones y velocidades de manera aleatoria dentro de un rango específico para acotar el espacio de búsqueda.

- **Paso 2: Entrenamiento.** La LS-SVM es entrenada con los parámetros incluidos en una partícula. La estructura de la partícula estará condicionada al tipo de Kernel a utilizar en la SVM. La tabla[1] 5.1 muestra los posibles parámetros que la partícula podría representar [Wu et al., 2009]:

- **Paso 3: Obtención del Valor Fitness**. Este valor es calculado a través de la evaluación de la función fitness. Esta función, recibe como entrada el resultado de los valores pronosticados por LS-SVM en su fase de entrenamiento con los parámetros representados por la partícula. Para el caso específico de este modelo, las funciones

[1]Nota: – denota que no se requiere el parámetro; mientras que *gamma*, d, t, C, σ^2 son los parámetros específicos del Kernel a utilizar.

fitness consideradas corresponden a el Error Cuadrático Medio y a la Validación Cruzada.

- **Paso 4: Actualización.** Se actualiza la mejor posición de la partícula y del enjambre según el valor fitness. Posteriormente cada una de las partículas se somete al proceso de actualización tanto en velocidad como en posición.
- **Paso 5: Finalización**. Se verifica si se cumple con el criterio de término definido, de no ocurrir lo anterior, se vuelve al paso 2, en caso contrario continuar.
- **Paso 6: Obtención de los Parámetros Óptimos.** Una vez que se ha terminado con la ejecución del algoritmo PSO, se obtiene la partícula que mejor representa la solución del problema, es decir, cuyo valor fitness presentó el menor valor de la población.
- **Paso 7: Evaluación del Modelo.** Luego de obtener los parámetros óptimos del modelo, éste debe someterse al proceso de evaluación con los datos pertenecientes a la etapa de validación, a través del uso de métricas definidas en 5.4.

Tipo Kernel	**Parámetro 1**	**Parámetro 2**
Lineal	*gamma*	-
Polinomial	d	t
RBF	C	σ^2

Tabla 5.1: *Parámetros de la partícula con respecto al Kernel seleccionado.*

5.6 Descripción y Tratamiento de los Datos

Los datos utilizados corresponden a los valores asociados a los desembarques totales registrados para la especie anchoveta de la zona norte de Chile. Se han considerado las capturas desde Enero de 1963 hasta Diciembre de 2007, alcanzando un total de 384 registros.

Para el modelo propuesto, los datos asociados al recurso pelágico fueron en primera instancia normalizados con la ecuación (5.6.1), para luego ser desfasados dentro del rango de 4 a 12 meses, todo esto con el objeto de obtener la cantidad de meses previos que mejor represente a la serie de datos en estudio.

Posteriormente, los datos fueron divididos en dos grupos: la primera porción para la etapa de entrenamiento y la restante para la fase de validación. El tamaño de la muestra fue evaluada dentro del rango del 50 % al 90 % del total de los datos. Ambos grupos fueron separados en entradas y salidas, siendo las primeras de ellas suavizadas, con el fin de

eliminar el ruido y así lograr una clara observación de la tendencia de los datos. Para este caso, se utilizó un promedio móvil de 3 meses.

$$y(t) = \frac{y(t)}{max(t)} \tag{5.6.1}$$

5.7 Proceso de Obtención de Parámetros

El modelo LS-SVM + PSO — para cualquiera de sus variantes —, cuenta con una gran cantidad de parámetros, por lo que es necesario, plantear una metodología que permita obtener la mejor configuración y por ende el mejor modelo. A continuación se describirá el proceso de obtención de parámetros para cada uno de los modelos a evaluar.

1. **Establecer los parámetros iniciales para el algoritmo PSO:** A partir de esta base, se espera determinar la mejor configuración de cada variante del algoritmo PSO. Los parámetros que se han definido como determinantes a la hora obtener una buena calidad en los resultados son: el número de partículas y el número de iteraciones del algoritmo — criterio de término —.

2. **Elegir la mejor función finess:** La función fitness cumple un rol fundamental a la hora de medir la calidad de las partículas. Es por esto que se medirá las funciones fitness expuestas en 5.3, con el objeto de conseguir la mejor.

3. **Restringir el espacio de búsqueda:** Si el algoritmo PSO abarca un espacio de búsqueda muy extenso, es posible que demore en converger hacia un valor óptimo. generando una baja precisión en los resultados. La idea es iniciar el algoritmo PSO en un buen espacio de búsqueda.

4. **Obtener el mejor desfase de los datos:** A partir de los parámetros de la etapa anterior, se procederá a ejecutar el algoritmo PSO en su versión original. La idea es encontrar y analizar el porcentaje de varianza explicada por cada desfase dentro del rango de 6 a 12 meses. El criterio de selección del mejor desfase, se hará a partir de la mejor media obtenida en cuanto al coeficiente de determinación se refiere.

5. **Conseguir el mejor tamaño de la muestra:** En función de lo descrito en la sección 5.6, se procederá a evaluar el comportamiento de los datos, con el objetivo de crear un modelo que generalice bien el problema. La idea es evitar el sobreajuste a los datos de entrenamiento. Este proceso se realizó con el desfase obtenido en la etapa anterior y bajo el mismo criterio de selección.

6. **Obtener el tamaño del enjambre:** En función del mejor desfase obtenido, se procederá a ejecutar cada variante del algoritmo PSO por separado. La dinámica en esta etapa será la de variar el tamaño del enjambre. La idea es encontrar un buen número de partículas que permitan explorar y explotar de mejor manera el espacio de

soluciones. El criterio de selección del mejor enjambre, se hará de la misma manera que en las etapas previas, la mejor media de la varianza explicada.

7. **Obtener el número de iteraciones:** Un factor importante para cualquier algoritmo PSO, es su criterio de término, ya que si este valor es muy pequeño, es posible que no se puedan encontrar buenas soluciones, por otro lado, si el número es muy alto, puede ocurrir que los parámetros de la LS-SVM se sobreajusten mucho al problema, generando soluciones indeseadas. La idea es hacer una compensación entre la calidad de la soluciones y el tiempo de ejecución del algoritmo, con esto se espera obtener un modelo que requiera poco tiempo en ser calibrado y que a su vez obtenga soluciones que cumplan la calidad esperada.

8. **Obtener el mejor modelo:** Ya encontrados los parámetros críticos para cada algoritmo PSO. Se procederá a elegir la mejor variante y ejecutar un número determinado de repeticiones con el afán de encontrar el mejor de todos los modelos, este proceso se hará a través de las métricas de evaluación. — ver 5.4 —

Capítulo 6

Modelo Propuesto y Análisis de Resultados

6.1 Introducción

En este capítulo se realizará el análisis y discusión de resultados de los modelos obtenidos para el pronóstico de anchovetas de la zona norte de Chile. Estos modelos obedecen a la estructura general descrita en la sección 5.5, es decir, a una *Máquina de Soporte Vectorial* con optimización de parámetros a través de tres variantes del algoritmo *Particle Swarm Optimization*: IWPSO, LAPSO y AMPSO — ver 4.6 —.

El proceso de obtención del mejor modelo se hizo por medio de las secuencias descritas en el capítulo anterior — ver 5.7 —.

6.2 Modelos Propuestos LS-SVM + PSO

Como se describe en la introducción, la configuración de los parámetros internos de la LS-SVM se desarrollará por medio de tres variantes del algoritmo PSO:

- LS-SVM + IWPSO.
- LS-SVM + LAPSO.
- LS-SVM + AMPSO.

Para todos ellos se ha utilizado la función Kernel de base radial — RBF —, debido a que ofrece un muy buen comportamiento para la mayoría de los problemas.

La etapa de entrenamiento se inicia con la creación de una población inicial de partículas, de 2 dimensiones cada una, éstas hospedarán los valores de los parámetros

a optimizar para la SVM, la primera componente hace referencia al parámetro de regularización C, encargado de compensar la maximización del margen y la minimización del error de entrenamiento, mientras la segunda representa el parámetro σ^2 asociado al Kernel RBF a utilizar — ver tabla 5.1 —.

Cuando el algoritmo comienza a ejecutarse, por cada proceso iterativo, se deberá evaluar la calidad de cada una de las partículas a través de la función fitness utilizada, ésta función permite determinar el grado de adecuación de las partículas sobre el problema. Una vez que el criterio de término se haya cumplido, la mejor partícula obtenida se hará cargo de representar los parámetros óptimos para la SVM.

6.3 Parámetros Iniciales y Pre-proceso de los Datos

En esta sección se describirá las primeras cuatro etapas del proceso de obtención de parámetros y pre-proceso de los datos — ver 5.7 y 5.6—. Por una parte se buscará encontrar los parámetros iniciales de cada algoritmo PSO y su función fitness, y por otro el desfase de los datos y el tamaño de la muestra que mejor represente el problema.

6.3.1 Parámetros Iniciales PSO

A través de exámenes previos, se pudo vislumbrar el siguiente estado inicial para cada algoritmo PSO:

Parámetro	Valor Asociado
Tamaño Enjambre	10
Número de Iteraciones	10
Rango Parámetro C	[1 ; 100000]
Rango Parámetro σ^2	[1 ; 5000]
Factor Cognitivo ρ_1	1.8
Factor Social ρ_2	2.4

Tabla 6.1: *Parámetros Iniciales del Algoritmo PSO.*

Los parámetros específicos de cada variante fueron asignados con los siguientes valores:

- IWPSO y AMPSO: Ambas variantes poseen el factor de inercia w — ver ecuación (4.6.2) —. Este factor tiene la particularidad de que a medida que las iteraciones del algoritmo van en aumento este irá decreciendo a través del tiempo dentro del rango establecido. – ver ecuación (4.6.3) —, garantizando una mayor explotación de la vecindad en la últimas iteraciones. Los valores asignados fueron:

- $w_{min} = 0{,}4$.
- $w_{max} = 0{,}9$.

- LAPSO: Para evitar una rápida convergencia es que se ha optado por asignar los valores recomendados en [Yisu et al., 2008], los cuales son:

  - $D_{min} = 0{,}4$.
  - $D_{max} = 0{,}95$.

Cabe destacar que todas las variantes se les aplico un control de velocidad — V_{max} —, para un equilibrar el movimiento de las partículas en el espacio de búsqueda.

6.3.2 Función Fitness

Ya teniendo los parámetros base del algoritmo PSO, se procedió a evaluar la calidad de las dos funciones fitness consideradas. Por un lado la función MSE y por el otro la de Validación Cruzada.

Para llevar a cabo este estudio, se realizaron 5 repeticiones utilizando el algoritmo PSO original, con el fin de evitar sesgos entre variantes. A su vez el porcentaje de entrenamiento fue de un 50 % y el número de desfases fue de 6 meses.

Fitness	Máximo	Mínimo	Media	Desv. Estándar
MSE	0.870776867009669	0.588600536964876	0.813083809231569	0.125506713837874
Validación Cruzada	**0.866933682026311**	**0.844244915261242**	**0.855857301308592**	**0.008884497350166**

Tabla 6.2: *Resumen R^2 por función Fitness.*

Como se puede apreciar en la tabla 6.2, la función de validación Cruzada presenta mejor comportamiento, ya que la media del R^2 fue superior en mas de 4 puntos porcentuales por sobre la media entregada por la función MSE. Además su desviación estándar es bastante inferior, lo que garantiza buena estabilidad en los resultados.

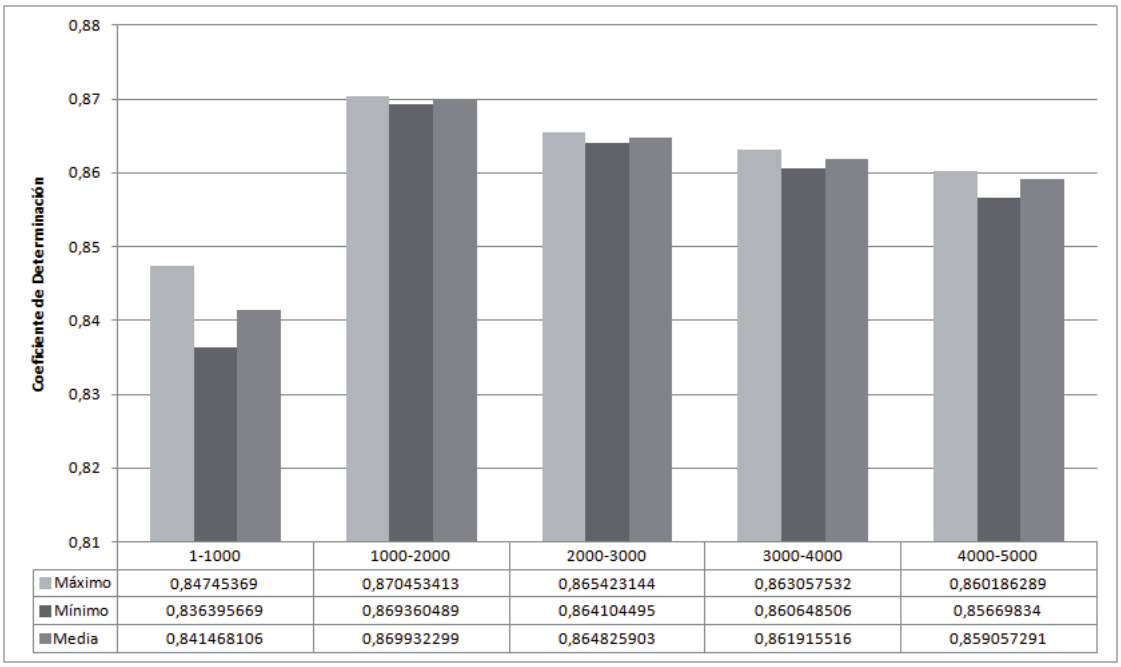
Por lo tanto el fitness seleccionado para el resto del proceso ha sido la función de Validación Cruzada.

6.3.3 Espacio de Búsqueda

Las SVMs son muy susceptibles a los parámetros de su kernel, ya que en gran medida determinan el grado de generalización o sobreajuste del modelo en la etapa de entrenamiento. Es por esto, que es de suma importancia establecer un buen espacio de

búsqueda inicial para las partículas del algoritmo PSO, ya que esto puede garantizar una buena calidad en los resultados y a su vez disminuir considerablemente el tiempo de entrenamiento de la máquina.

Para el caso particular en estudio, se pudo vislumbrar una fuerte relación entre la calidad de los resultados y los valores del parámetro σ^2 del Kernel seleccionado — RBF —. La figura 6.1 expone la variación del coeficiente de determinación por cada rango establecido para el parámetro en cuestión.



	1-1000	1000-2000	2000-3000	3000-4000	4000-5000
Máximo	0,84745369	0,870453413	0,865423144	0,863057532	0,860186289
Mínimo	0,836395669	0,869360489	0,864104495	0,860648506	0,85669834
Media	0,841468106	0,869932299	0,864825903	0,861915516	0,859057291

Figura 6.1: *Variación R^2 por rango de σ^2.*

Como se puede ver, existen óptimos locales de baja calidad dentro del rango [1-1000], no así en el resto, sobre todo en el segundo rango, en donde el valor observado fue de un 86,99 % de R^2. Bajo esto resultados, se ha decidido descartar dentro del espacio de búsqueda el primer rango de valores.

6.3.4 Número de meses de desfase

El número de desfase al igual que el proceso anterior, se realizó utilizando el algoritmo PSO original. Los meses fueron evaluados dentro del rango de 6 a 12 meses, ejecutando 5 repeticiones por cada uno de estos desfases y utilizando la función fitness ya seleccionada.

Desfase	Máximo	Mínimo	Media	Desv. Estándar
6	0.8693604889686	0.87045341266161	0.86993229923813	0.000501909287181
7	0.85636813844763	0.85395420040893	0.85535379915731	0.001078000232622
8	0.87434697613486	0.87193185175368	0.87315478119261	0.000864215550489
9	0.89532590717369	0.89339697403848	0.89453378673232	0.000718881169634
10	0.90005065214991	0.89688001373311	0.89866616468172	0.001285296023865
11	0.90991728833154	0.90798461540532	0.90931603323251	0.000783989793952
12	**0.92323307777536**	**0.91757056254563**	**0.92035130897637**	**0.002565202884544**

Tabla 6.3: *Resumen R^2 por Desfase.*

La tabla 6.3 muestra la variación de la media del coeficiente de determinación por cada uno de los desfases. Como se puede observar, existe una clara y directa correlación entre la calidad de los resultados y el mes de desfase, ya que a medida que aumentan los meses, los resultados van mejorando considerablemente.

El desfase de 12 meses presenta el mejor promedio de R^2, cuyo valor fue cercano al 92,04 %. Por lo tanto, es claro decir que el mejor desfase corresponde al de 12 meses, por lo cual fue el valor seleccionado para el resto de las pruebas.

6.3.5 Tamaño de la muestra

La última etapa del pre-proceso de los datos corresponde a la selección del mejor tamaño de la muestra. Es por ello que se realizó una variación entre el 50 % y el 90 % del total de datos para la etapa de entrenamiento. Al igual que en todos los otros procesos se ha utilizado el algoritmo PSO en su forma original.

% Muestra	Máximo	Mínimo	Media	Desv. Estándar
50	0.91757056254563	0.92323307777536	0.92035130897637	0.002565202884544
60	0.93634681136057	0.93829968252345	0.93752379275128	0.000766250510094
70	**0.94072853273285**	**0.94543841502031**	**0.94373399009238**	**0.001879863275536**
80	0.90307667813451	0.9063021598192	0.90444551136291	0.001233732491478
90	0.91256113926586	0.91492729842276	0.91384055871108	0.001016566508735

Tabla 6.4: *Resumen R^2 por Porcentaje de Muestra.*

La tabla 6.4 expone la variación del coeficiente de determinación por cada porción de muestra seleccionada. No existe un patrón que determine de manera clara la relación entre la cantidad de datos y la calidad de los resultados, pero si una notoria diferencia entre el porcentaje que generaliza mas el problema, el cual corresponde al 70 % del total de datos. En esa muestra se observó un valor cercano al 94,37 % de la varianza explicada, superando en mas de un punto porcentual al mas cercano que corresponde al 60 %.

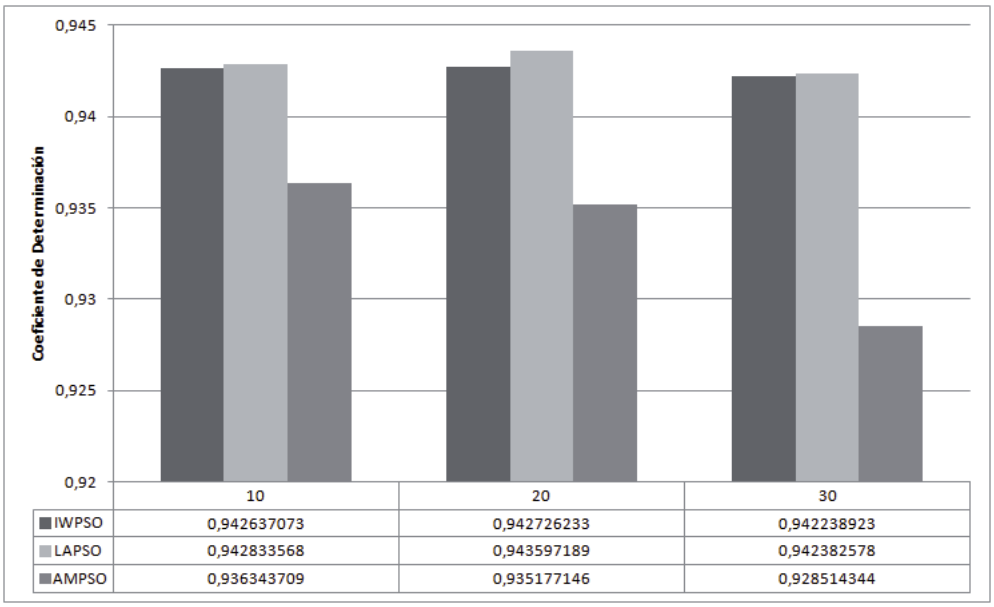
Por lo tanto la muestra seleccionada corresponde al 70 % de los datos, que corresponde a un total de 267 registros.

6.4 Obtención de Parámetros y Selección Variante PSO

Teniendo los resultados finales del pre-proceso de los datos y la configuración inicial de cada una de las variantes del algoritmo PSO, se procederá a exponer a continuación los resultados obtenidos en las pruebas para cada una de las modificaciones PSO.

6.4.1 Tamaño del Enjambre

El tamaño del enjambre fue evaluado a través de su número de partículas. El rango considerado fue de 10 a 30 partículas y fueron evaluados en cada una de las variantes del algoritmo PSO.



	10	20	30
IWPSO	0,942637073	0,942726233	0,942238923
LAPSO	0,942833568	0,943597189	0,942382578
AMPSO	0,936343709	0,935177146	0,928514344

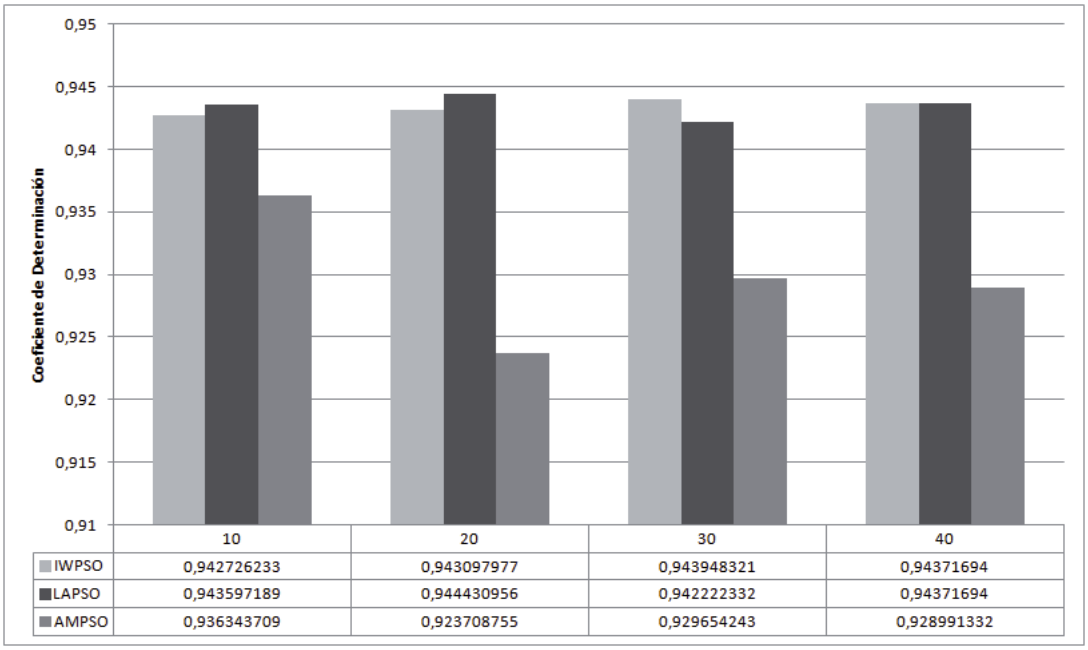
Figura 6.2: *Variación R^2 por Número de Partículas.*

La figura 6.2 muestra la variación que presenta la media de R^2 por cada valor de número de partículas. Se puede apreciar que las variantes IWPSO y LAPSO presentan un rendimiento similar, en donde las partículas de tamaño 20 arrojan un mejor promedio para ambos casos. Situación totalmente diferente ocurre con la variante AMPSO, ya que a medida que aumenta el número de partículas, la calidad de resultados se ve disminuida considerablemente, además en contraste con las otras variaciones, su nivel muestra una inferioridad de casi 1 punto porcentual.

Por lo tanto, el número de partículas seleccionadas para IWPSO y LAPSO es de 20 y para AMPSO de 10.

6.4.2 Número de Iteraciones

El criterio de parada para los algoritmos fue evaluado dentro del rango de 10 y 40 iteraciones, en donde los resultados se ven reflejados en la figura 6.3.



	10	20	30	40
IWPSO	0,942726233	0,943097977	0,943948321	0,94371694
LAPSO	0,943597189	0,944430956	0,942222332	0,94371694
AMPSO	0,936343709	0,923708755	0,929654243	0,928991332

Figura 6.3: *Variación R^2 por Número de Iteraciones.*

Los resultados observados muestran poca variación para las modificaciones IWPSO y LAPSO, sin embargo esta última presentó la mejor media, con un valor cercano al 94,44% en las 20 iteraciones. Para el caso de la variante IWPSO su máximo valor lo presentó en las 30 iteraciones con un valor del 94,39% de la varianza explicada. La variante AMPSO, no pudo mejorar la calidad de sus resultados, por lo que se descarta completamente en la selección del mejor algoritmo.

Finalmente se puede concluir que el algoritmo LAPSO presenta mejores resultados que el resto de sus competidores y por ende éste será el algoritmo seleccionado.

6.5 Selección del Mejor Modelo

En base a los estudios y resultados previos se pudo concluir que el mejor modelo obtenido hasta el momento correspondió al LS-SVM + LAPSO, sin embargo es necesario realizar un estudio mas acabado de la bondad que este modelo puede entregar.

Para lograr el cometido anteriormente mencionado, es que se decidió ejecutar 20 repeticiones de la mejor configuración del algoritmo LAPSO, con el afán de seleccionar el mejor modelo y los valores óptimos para la SVM — C y σ^2—.

La tabla 6.5 entrega el resumen de las métricas de los modelos obtenidos después de realizar las 20 ejecuciones. Acá se puede observar que si bien es cierto se logran resultados de variabilidad baja, aún sigue existiendo una gran brecha entre los máximos y mínimos, por lo que la precisión del modelo sigue siendo baja. Para poder solucionar este problema y garantizar soluciones dentro de un rango pequeño, se ha decidido replantear el espacio de búsqueda acotándolo a sólo los mejores óptimos, es decir, disminuyendo las cotas superiores e inferiores del parámetro σ^2 del kernel.

	Máximo	**Mínimo**	**Media**	**Desv. Estándar**
MAE	0.029056043337616	0.02771827450282	0.028219970925078	0.00044390111631
MSE	0.001431069716103	0.001240473455835	0.001310768475513	0.000058392051338
RMSE	0.037829482101968	0.035220355702841	0.0361961063419	0.000801551915182
MAPE	61.7508954965531	52.0716050691505	55.8381442996152	2.50718150397682
MASE	0.26545658403239	0.25323470161765	0.25781821000957	0.004055489338818
R^2	0.94611732914954	0.93996397087693	0.94397407887801	0.001920168405456

Tabla 6.5: *Resumen Métricas Modelo Previo.*

La configuración final ha quedado como indica la tabla 6.6. Aquí se puede apreciar que el valores de σ^2 estarán dentro del rango de 1500 y 3000, ya que de lo contrario los resultados se verán afectados y por ende un sobreajuste del modelo. Con la nueva configuración y final del modelo, se realizó la última etapa de pruebas, tal como indica la tabla 6.7.

Parámetro	**Valor Asociado**
Meses de Desfase	12
Tamaño Muestra	70 %
Tamaño Enjambre	20
Número de Iteraciones	20
Rango Parámetro C	[1 ; 100000]
Rango Parámetro σ^2	[1500 ; 3000]
Factor Cognitivo ρ_1	1.8
Factor Social ρ_2	2.4
Función Fitness	Validación Cruzada
D_{min}	0.4
D_{max}	0.95

Tabla 6.6: *Configuración Final LSSVM + LAPSO.*

Repetición	Parámetro C	Parámetro σ^2	MAE	MSE	RMSE	MAPE	MASE	R^2	Tiempo
1	98811.33	2089.0	0.027754	0.0012425	0.035248	60.126	0.25356	0.94625	00:00:23
2	94688.85	1836.9	0.027809	0.0012526	0.035392	58.725	0.25406	0.94596	00:00:22
3	79892.67	1853.6	0.027755	0.0012438	0.035267	59.795	0.25357	0.94618	00:00:28
4	76453.35	1577.6	0.027822	0.0012590	0.035483	57.962	0.25418	0.94573	00:01:05
5	57023.88	1572.2	0.027744	0.0012444	0.035277	59.547	0.25347	0.94610	00:01:04
6	81633.16	1790.6	0.027778	0.0012474	0.035319	59.235	0.25378	0.94609	00:01:04
7	**89690.96**	**2048.4**	**0.027738**	**0.0012410**	**0.035228**	**60.414**	**0.25341**	**0.94626**	**00:01:05**
8	85876.70	1771.4	0.027798	0.0012512	0.035372	58.811	0.25396	0.94599	00:01:02
9	80030.76	2521.9	0.027754	0.0012448	0.035282	63.608	0.25356	0.94585	00:01:02
10	99531.03	1577.3	0.028079	0.0012850	0.035847	56.612	0.25653	0.94493	00:01:00
11	80625.27	1527.4	0.027895	0.0012690	0.035623	57.300	0.25485	0.94542	00:00:59
12	77050.34	1803.4	0.027759	0.0012445	0.035278	59.647	0.25360	0.94616	00:01:00
13	98689.49	1673.8	0.027926	0.0012709	0.035650	57.343	0.25513	0.94539	00:01:11
14	94415.88	1696.2	0.027862	0.0012643	0.035557	57.721	0.25455	0.94559	00:01:04
15	71015.83	2094.7	0.027713	0.0012399	0.035212	61.948	0.25319	0.94615	00:01:07
16	84939.76	1515.8	0.027963	0.0012758	0.035718	56.947	0.25547	0.94521	00:01:06
17	66509.57	1992.7	0.027710	0.0012399	0.035212	61.674	0.25316	0.94615	00:01:05
18	34425.81	1888.3	0.027987	0.0012641	0.035555	64.316	0.25569	0.94485	00:01:06
19	97268.55	1675.8	0.027907	0.0012693	0.035627	57.430	0.25496	0.94544	00:01:03
20	96237.89	1776.3	0.027833	0.0012583	0.035473	58.195	0.25428	0.94579	00:01:03

Tabla 6.7: *Tabla de resultados de la mejor configuración.*

El mejor modelo obtenido fue en la repetición número 7, con un porcentaje de varianza explicada del 94,63 % y tomo un poco mas de 1 minuto en ser calibrado. Además se puede apreciar que la precisión del modelo disminuyó considerablemente, tal y como se puede apreciar en la tabla 6.8

	Máximo	Mínimo	Media	Desv. Estándar
MAE	0.028078697242159	0.02771041878416	0.027829203519957	0.000101319956709
MSE	0.001284979621525	0.001239869767174	0.001255388181595	0.000013602759877
RMSE	0.035846612413514	0.035211784492896	0.035430964955447	0.000191679340135
MAPE	64.3158200729168	56.6122443293077	59.3677565863333	2.16178923743551
MASE	0.25652753086081	0.2531629316173	0.25424815130236	0.000925661119433
R^2	0.94626249874186	0.94485365770493	0.94577449288521	0.000438897066804

Tabla 6.8: *Resumen Métricas Mejor Configuración.*

La figura 6.4 muestra la dispersión del conjunto de valores obtenidos para R^2. Para las pruebas realizadas el modelo seleccionado presentó un 75 % de precisión.

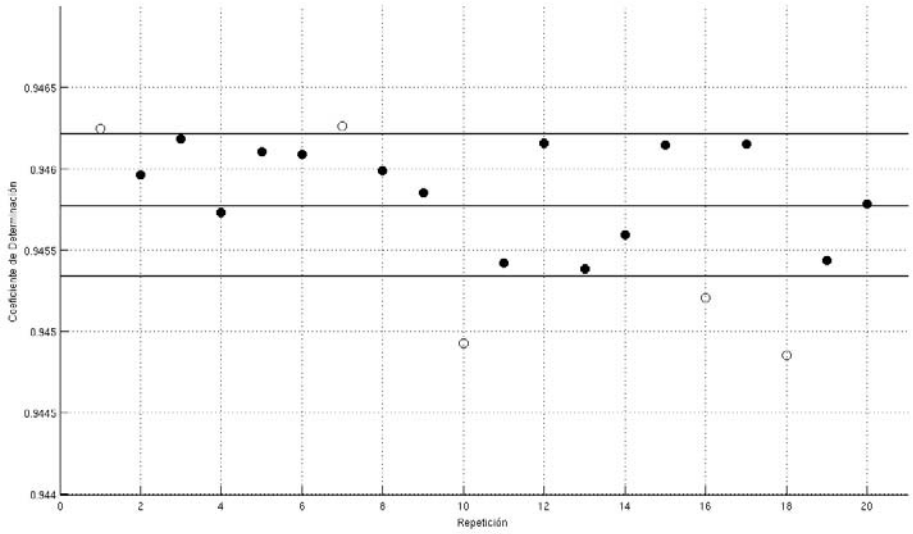


Figura 6.4: *Precisión del Modelo.*

6.6 Modelo Final

Finalmente se puede exponer la configuración final del mejor modelo LS-SVM + LAPSO para la especie Anchoveta de la zona norte de Chile — ver tabla 6.9 —. Las figura 6.5 muestra los desembarques pronosticados por el modelo, mientras que la figura 6.6 representa la dispersión de estos mismos.

Componente/Métrica	**Valor**
Parámetro SVM C	89690.96
Parámetro SVM σ^2	2048.4
R^2	0,9463
Tiempo	01:05

Tabla 6.9: *Mejor modelo LS-SVM + LAPSO para la especie Anchoveta.*

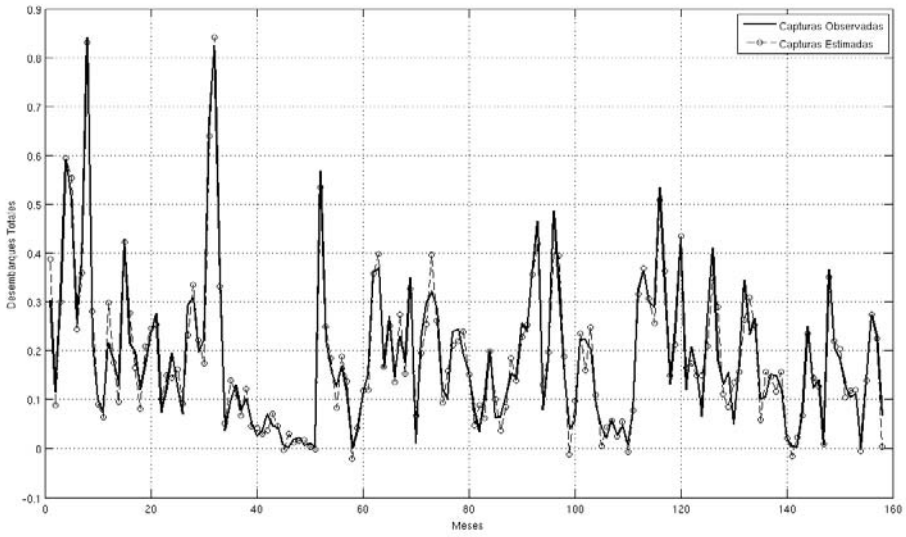


Figura 6.5: *Desembarques Observados vs Desembarques Estimados.*

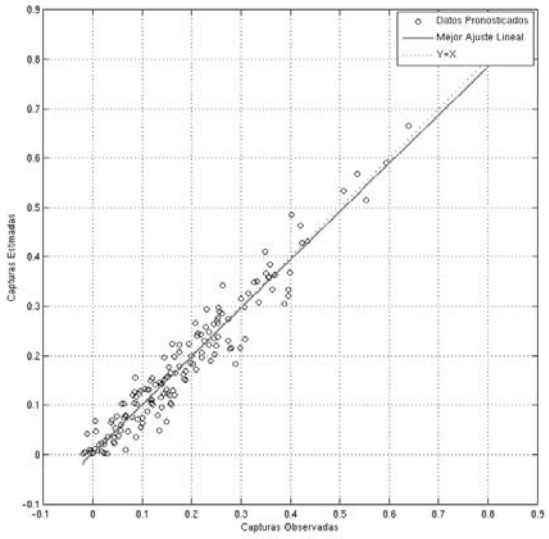


Figura 6.6: *Dispersión entre los desembarques Observados y Estimados para la Anchoveta.*

Capítulo 7

Conclusiones

Las máquinas de soporte vectorial — SVM — debido a que su formulación encarna el principio minimización del riesgo estructural, tienen la capacidad de trabajar con un conjunto pequeño de datos de entrenamiento. Si bien es cierto que las SVMs nacieron para dar solución al problema de clasificación, gracias a la adición de una nueva componente denominada función de pérdida — *loss function* — y a la integración del concepto Kernel para problemas no lineales, han resultado viables para los problemas de regresión. Una de las debilidades de las SVMs es la gran susceptibilidad de sus parámetros de ajuste, ya que una pequeña variación de estos mismos, pueden generar modelos altamente inestables y por sobretodo muy lejos del óptimo. Para apalear este problema, resulta imperante recurrir a técnicas de optimización.

El algoritmo de optimización basado en enjambre de partículas — PSO —, presenta un funcionamiento bastante definido y robusto, por lo que resulta ser una buena alternativa para su incorporación en la creación de un modelo en el cual los parámetros suelen ser complicados de hallar. Esta técnica busca dentro del espacio de solución, la más óptima. La existencia de varias soluciones al problema, hacen que la tarea de encontrar el óptimo global sea más difícil, porque existe la posibilidad de que el algoritmo converja hacia un óptimo local y no pueda salir de ese subespacio de soluciones. Para esto, es que se han definido algunas variaciones de su funcionamiento, con el afán de recorrer todo el espacio de solución en busca del óptimo global sin caer antes en uno local. Esto lo realizan a través de técnicas que buscan ejercer una buena compensación entre la exploración y la explotación del espacio de soluciones.

Por medio de las técnicas LS-SVM y PSO, se diseñó un modelo que fuese capaz de pronosticar el nivel de captura de achovetas de la zona norte de Chile, a través de reiterados procesos de prueba y validación. El proceso de estimación de parámetros de la SVM, se realizó mediante diferentes variantes del algoritmo PSO: IWPSO, LAPSO, AMPSO. La función fitness Cross Validation presentó mejores resultados que el fitness MSE, arrojando una media de R^2 superior por sobre los 4 puntos porcentuales con respecto a su competidora, a su vez, cabe mencionar que la desviación estándar experimentada fue

bastante inferior para esta función fitness. Con respecto a los datos, se logró apreciar que utilizando un tamaño de muestra del 70 % y un desfase de 12 meses, la integridad de los modelos generados presentaron mejores resultados.

Para la selección del mejor modelo, se utilizaron diversas métricas de evaluación, a partir de ellas, se logró concluir que el el modelo LS-SVM + LAPSO, era capaz de representar de mejor manera los datos del problema, arrojando mejores resultados por sobre las otras variantes — IWPSO y AMPSO —. La ganancia del mejor modelo fue de un 0,25 % con respecto al mejor modelo IWPSO y de un 0,75 % para el caso del AMPSO. Por otro lado, la estimación de la desviación estándar del error de predicción fue de un 3,5 %, mientras que el MAPE fue de un 60,41 %.

Finalmente, se puede concluir que las SVMs son capaces de presentar un buen modelo de pronóstico para la captura de Anchovetas de la zona norte de Chile, ya que el modelo propuesto presenta un $94,63\,\%$ de la varianza explicada.

Referencias

[Bichop, 1995] Bichop, C. (1995). *Neural Network for Pattern Recognition*. Oxford University Press.

[Burges, 1998] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining And Knowledge Discovery*, pages 2:1–17.

[Cherkassky and Mulier, 1998] Cherkassky, V. and Mulier, F. (1998). *Learning from data: concepts, theory, and methods*. John Willey and Sons, New York.

[Clerc, 1999] Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proceedings of 1999 Congress on Evolutionary Computation, 1999. CEC 99*, 3:–1957.

[Clerc and Kennedy, 2002] Clerc, M. and Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evolutionary Computation*, pages 6(1):58–73.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. . (1995). Machine learning. *Support vector networks*, pages 20:273–297.

[De Brabanter et al., 2010] De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle, J., and Suykens, J. A. K. (2010). Ls-svmlab toolbox user's guide : version 1.7. Technical Report 10-146, Uppsala University, Division of Systems and Control.

[Donoso, 2010] Donoso, W. D. (2010). *Informe Sectorial Sector Pesca y Salmonicultura*. Fitch Ratings.

[Fletcher, 1987] Fletcher, R. (1987). *Practical methods of optimization*. John Wiley and Sons, Inc, 2 edition.

[Gardner, 1975] Gardner, M. (1975). Mathematical games: on the remarkable császár polyhedron and its applications in problem solving. *En Sci. Amer.*, pages 102–107.

[Hyndman and Koehler, 2006] Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688.

[Karush, 1939] Karush, W. (1939). *Minimal of Functions of Several Variables with Inequalities as Side Constraints*. M.Sc. Dissertation.Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois.

[Kennedy, 1999] Kennedy, J. (1999). Small world and mega-minds: Effects of neighborhood topology on particle swarm performance. *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1931–1938.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *In Proceedings of IEEE International Conference on Neural Networks, volume 4*, pages 1942–1948.

[Kennedy and Mendes, 2003] Kennedy, J. and Mendes, R. (2003). Neighborhood topologies in fully-informed and best-of-neighborhood particle swarm. *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications*, pages 45–50.

[Kennedy, 2001] Kennedy, J. y Eberhart, R. (2001). *Swarm Intelligence*. Morgan Kaufmann Publishers.

[Pant et al., 2008a] Pant, M., Thangaraj, R., and Abraham, A. (2008a). A new quantum behaved particle swarm optimization. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 87–94, New York, NY, USA. ACM.

[Pant et al., 2008b] Pant, M., Thangaraj, R., and Abraham, A. (2008b). Particle swarm optimization using adaptive mutation.

[Peer and Engelbrecht, 2003] Peer, E. S., v. d. B. F. and Engelbrecht, A. P. (2003). Using neighborhoods with the guaranteed convergence pso. *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 235–242.

[Schölkopf and Smola., 2001] Schölkopf, B. and Smola., A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA.

[Sectorial, 2010] Sectorial, D. A. (2010). *Informe Sectorial de Pesca y Acuicultura*.

[Sedighizadeh and Masehian, 2009] Sedighizadeh, D. and Masehian, E. (2009). Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5):486–502.

[Shi and Eberhart, 1998] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pages 69–73.

[Shi and Eberhart, 1999] Shi, Y. and Eberhart, R. (1999). Empirical study of particle swarm optimization. *Proceedings of 1999 Congress on Evolutionary Computation, 1999*, 3:–1950.

[Smola, 1998] Smola, A. (1998). *Learning with Kernels*. Department Computer Sciense, Technical University Berlin, Germany.

[Smola and Schölkopf, 1998] Smola, A. J. and Schölkopf, B. (1998). On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Technical Report 1064, GMD FIRST*, pages 22:211–231.

[S.R. Gunn and Bossley, 1997] S.R. Gunn, M. B. and Bossley, K. (1997). *Network performance assessment for neurofuzzy data modelling*. In X. Liu, P. Cohen, and M. Berthold.

[Suykens and Vandewalle, 1999] Suykens, J. and Vandewalle, J. (1999). *Least Squares Support Vector Machine Classifiers.* Neur. Proc. Lett.

[T. Poggio and Koch., 1985] T. Poggio, V. T. and Koch., C. (1985). *Computational vision and regularization theory.* Nature.

[Tikhonov and Arsenin, 1977] Tikhonov, A. and Arsenin (1977). *Solution of ill-posed problems*. W.H. Winston, Washington, DC.

[Vapnik, 1995] Vapnik (1995). *Statistical learning theory*. John Willey and Sons, New York.

[Vapnik, 1998] Vapnik (1998). *The nature of statistical learning theory*. Springer Verlag, New York.

[Vapnik and Smola, 1997] Vapnik, S. G. and Smola, A. (1997). *Support vector method for function approximation, regression estimation, and signal processing*. In M. Mozer, M. Jordan, and T. Petsche, editors.

[Wu et al., 2009] Wu, C.-H., Tzeng, G.-H., and Lin, R.-H. (2009). A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Syst. Appl.*, 36:4725–4735.

[Yang et al., 2007] Yang, X., Yuan, J., Yuan, J., and Mao, H. (2007). A modified particle swarm optimizer with dynamic adaptation. *Applied Mathematics and Computation*, 189(2):1205–1213.

[Yisu et al., 2008] Yisu, J., Knowles, J., Hongmei, L., Yizeng, L., and Kell, D. B. (2008). The landscape adaptive particle swarm optimizer. *Applied Soft Computing*, 8(1).

[Zheng and Qian, 2003] Zheng, Y., M. L. Z. L. and Qian, J. (2003). Empirical study of particle swarm optimizer with an increasing inertia weight. *Proceedings of The 2003 Congress on Evolutionary Computation, 2003.*, 1:221–226.