Paralelización Eficiente en Métodos de Núcleo

Roberto Díaz Morales
Director: Ángel Navia Vázquez

Universidad Carlos III de Madrid, Departamento de Teoría de la Señal y Comunicaciones

10-02-2016

Universidad
Carlos III de Madrid

Toma de Decisión

Toma de decisión: Ante la observación de un dato $\mathbf{x} = \left[x_1, \ldots, x_p\right]^\top$ se realiza una elección entre un conjunto de hipótesis posibles $\Omega = \{\omega_1, \ldots, \omega_J\}$.

Se intenta obtener una función $f(\mathbf{x})$ que indique la hipótesis a aceptar.

Formas de obtención:

- Métodos analíticos:
  - Enfoque estadístico del problema.
  - La decisión tiene en cuenta el coste de la elección y la probabilidad a posteriori: $C(\omega_i|\mathbf{x}) = \sum_{j=1}^{J} c_{ij} P(\omega_j|\mathbf{x})$
  - Si se desconoce la probabilidad a posteriori se puede aplicar el teorema de Bayes: $P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{J} p(\mathbf{x}|\omega_j)P(\omega_j)}$
  - Si se desconoce la verosimilitud se puede aproximar a partir de muestras disponibles.
- Aprendizaje Máquina
  - No se tiene conocimiento estadístico del problema.
  - Se parte de un conjunto de muestras etiquetadas: $\{x_l, y_l\}_{l=1}^{n}$
  - Se utilizan funciones genéricas y se ajustan sus parámetros utilizando las muestras.

Aprendizaje Máquina

Tipos de Funciones:

- Modelos Lineales: Combinación lineal de los atributos de entrada: $f_j(\mathbf{x}) = \sum_{i=1}^{d}(\beta_{ji}x_i)$
- Métodos de Núcleo: Permiten crear de forma sencilla soluciones no lineales transformando el espacio de entrada en un espacio de alta dimensionalidad utilizando una función de núcleo. $f_j(\mathbf{x}) = \sum_{i=1}^{n}(\beta_{ji}K(\mathbf{x}, \mathbf{x_i}))$
- Redes Neuronales: Utilizan unidades de procesamiento no lineales organizadas por capas.
- Árboles: Los datos recorren una o varias estructuras en forma de árbol decidiendo en cada nodo por donde desplazarse.

Para obtener el valor de los parámetros internos se utiliza una función de coste. Mediante esta función el problema queda reducido a un problema de optimización.

Función de coste

Componente asociada al error:

- Error Cuadrático: $MSE = \frac{1}{L}\sum_{i=1}^{L}(\hat{y}_i - y_i)^2$
- Pérdida Logarítmica ("Log Loss"): $LL = \frac{1}{L}\sum_{i=1}^{L} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)$
- Pérdida de Bisagra ("Hinge Loss"): $HL = \frac{1}{L}\sum_{i=1}^{L} \text{máx}(0, (1 - y_i\hat{y}_i))$

Componente de Regularización:

- Norma l_1: $||\phi||_1$
- Norma l_2: $||\phi||_2$
- Red Elástica: Combinación lineal de las dos anteriores

Complejidad del modelo obtenido

Complejidad del modelo:

- Métodos Paramétricos: Tienen un conjunto fijo de parámetros independientemente del conjunto de entrenamiento utilizado. (Modelos lineales).
- No Paramétricos: La complejidad del modelo obtenido depende de los datos del conjunto de entrenamiento (SVMs, GPs).
- Semi-Paramétricos: Flexibles (GMMs).

Métodos de Núcleo

- Utilizan funciones de núcleo para modelar la no linealidad de los problemas.
- Pueden verse como un aprendizaje basado en instancias.
- Las Máquinas de Vectores Soporte (SVMs) esta muy extendidas en problemas de clasificación.
- Los Procesos Gaussianos (GPs) son muy utilizados en problemas de regresión.

Máquinas de Vectores Soporte

Parten de un **conjunto de entrenamiento** $\mathcal{D}$:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n$$

Utilizan la siguiente **función de coste**:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i$$

Sujeto a $y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, 2, ..., n$

$\xi_i \geq 0, i = 1, 2, ..., n$

Con el problema **dual** asociado:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

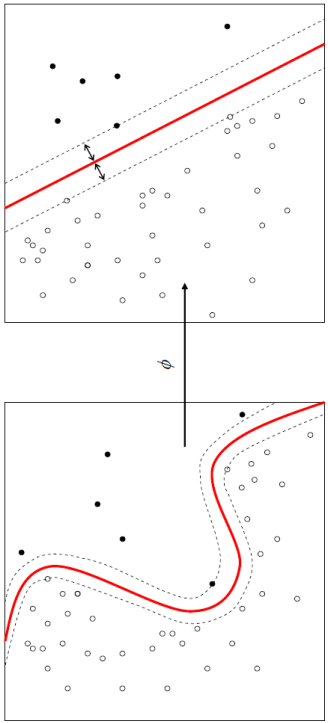
Sujeto a $\sum_{j=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C$

Soluciones no lineales mediante una **función de Kernel**:

$$\mathbf{x}_i \mathbf{x}_j \rightarrow K(\mathbf{x}_i, \mathbf{x}_j)$$

El **clasificador** tendrá la forma:

$$f(\mathbf{x}_i) = \sum_{j=1}^n y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + b$$



Procesos Gaussianos

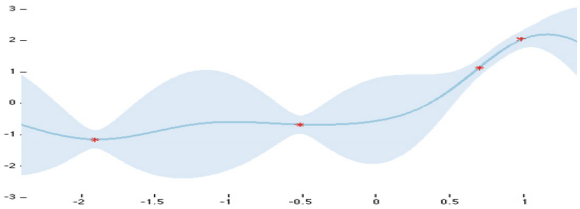
Un **Proceso Gaussiano** es un conjunto de variables aleatorias que tiene una distribución conjunta Gaussiana. Se define con su función de media y una función de núcleo a modo de covarianza:

$\mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$
$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
$cov(f(\mathbf{x}_1), f(\mathbf{x}_2)) = \mathbb{E}[(f(\mathbf{x}_1) - m(\mathbf{x}_1))(f(\mathbf{x}_2) - m(\mathbf{x}_2))] = K(\mathbf{x}_1, \mathbf{x}_2)$

En problemas de regresión($y_i = f(\mathbf{x}_i) + \xi_i$) permite hacer un enfoque bayesiano:

- Utilizando un prior GP sobre la función latente $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$
- Se da una probabilidad marginal de la salida: $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$
- Se obtiene una distribución a posteriori: $p(\mathbf{f}|\mathbf{y}) \sim \mathcal{N}(\mathbf{K}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, (\mathbf{K}^{-1} + \sigma^{-2}I)^{-1})$

Para realizar predicciones sobre nuevas muestras:
$p(f(\mathbf{x}_i)|\mathbf{y}, \mathbf{x}_i, \mathbf{X}, \theta, \sigma^2) = \mathcal{N}(\mathbf{k}_i^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, K(\mathbf{x}_i\mathbf{x}_i) - \mathbf{k}_i^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_i)$



Paralelización

El objetivo es dividir los problemas en tareas más pequeñas y utilizar **multiples recursos** de forma **simultánea** para resolverlas.

Ha surgido un gran interés en **adaptar técnicas clásicas** de aprendizaje automático a un nuevo escenario de paralización.

Tipos de paralización:

- **Distribuida:** Los sistemas trabajan de forma independiente y comunican resultados intermedios periódicamente utilizando un intercambio de mensajes.
- **Multinúcleo/Multiprocesador:** Actualmente los sistemas tienen varios microprocesadores y cada uno de ellos puede contener varios núcleos. Todos comparten la misma memoria RAM.
- **GPU:** Gran número de núcleos de capacidad muy reducida. Están pensados para que todos los núcleos realicen la misma operación sobre posiciones de memoria diferentes.

Motivación

Hechos

- Los **métodos de núcleo** como las **SVMs** o los **GPs** son métodos no paramétricos.
- El **coste computacional** asociado a su entrenamiento es $O(n^3)$.
- En los últimos años el tamaño de los conjuntos de entrenamiento ha crecido considerablemente.

Problema

- La falta de escalabilidad de los métodos de núcleo asociada al aumento de las bases de datos ha hecho que se haya reducido mucho su campo de acción.

Líneas de Investigación Actuales

- **Paralelización:** Utilización de multiples recursos de forma simultánea.
- Aproximaciones **Semi-Paramétricas:** Permiten fijar la complejidad de antemano.

Propuesta

- Reducir las limitaciones existentes en los métodos de núcleo haciendo uso de las lineas de investigación existentes.

Elección de algoritmos

Existen **numerosas alternativas** en cuanto a la resolución de métodos de núcleo tanto en su versión **no paramétrica** como **semi-paramétrica**.

No todas las opciones son idóneas para una implementación paralela.

Se ha seleccionado un algoritmo de resolución de **SVMs** y **GPs** y una aproximación semi-paramétrica de cada uno de ellos.

SVMs

Alternativas de Optimización:

- SMO: El más utilizado, divide el problema en los subproblemas del menor tamaño posible. Muchas iteraciones de coste computacional muy reducido.
- IPM: Pocas iteraciones pero de un coste computacional muy elevado.
- IRWLS: Compromiso intermedio entre número de iteraciones y coste computacional de cada iteración.

IRWLS:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i^n a_i e_i^2$$

Sujeto a:

$$w_i = \alpha_i y_i, \quad \sum_{j=0}^n w_i = 0,$$

$$e_i = y_i - \left(\sum_{j=1}^n w_j K(\mathbf{x}_i, \mathbf{x}_j) + b\right),$$

$$a_i = \begin{cases} 0, & y_i e_i \leq 0 \\ \frac{C}{e_i y_i}, & y_i e_i \geq 0 \end{cases}$$

SVMs

Modelo Completo:

- Utilizar todos los elementos del conjunto de entrenamiento en el sistema de mínimos cuadrados no es abordable.
- Se ha utilizado una separación de los datos en vectores soporte, vectores soporte fronterizos y muestras correctamente clasificadas. Sólo los vectores soporte no fronterizos se tienen en cuenta en el sistema.
- En cada iteración se ha seleccionado un conjunto de trabajo activo y uno inactivo. Los elementos del conjunto inactivo no se tienen en cuenta en el sistema de mínimos cuadrados.

Modelo Semi-Paramétrico:

- Se selección un conjunto de m elementos base $\{\mathbf{c}_i, \ldots, \mathbf{c}_m\}$ y $\mathbf{w}$ y se aproxima $\mathbf{w} \simeq \sum_{i=1}^{m} \beta_i \phi(\mathbf{c}_i)$ dando lugar a un clasificador de tamaño m:

  $f(\mathbf{x_i}) = \sum_{j=1}^{n} \beta_j K(\mathbf{x}_i, \mathbf{c}_j)$
- Los pesos $\boldsymbol{\beta}$ se obtienen entonces con el siguiente problema de optimización:

  $$\min_{\boldsymbol{\beta}} \frac{1}{2} \boldsymbol{\beta}^{\mathbf{T}} \mathbf{K_c} \boldsymbol{\beta} + C \sum_{i=1}^{n} \xi_i$$
- La selección de los elementos base se ha realizado con el algoritmo SGMA.

SVMs

Sparse Greedy Matrix Approximation

SGMA es una técnica muy utilizada para elegir los elementos de modelos semi-paramétricos.

Se puede aproximar una función de núcleo como:
$k(\mathbf{x}_i, .) = \sum_{j=1}^{m} \alpha_{i,j} k(\mathbf{c}_j, .)$

El error de esta combinación lineal es:
$Err(\alpha^{n,m}) = tr\mathbf{K} - \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_{i,j} k(\mathbf{x}_i, \mathbf{c}_j)$

El error de añadir un nuevo elemento a la base:
$$Err(\alpha^{n,m+1}) = Err(\alpha^{n,m}) - \underbrace{\eta^{-1}\|\mathbf{K}_{nm}\mathbf{z} - \mathbf{k}_{m+1,n}\|^2}_{\text{Descenso de Error}}$$
$(\mathbf{k}_{m+1,n})_i = k(\mathbf{c}_{m+1}, \mathbf{x}_i), \quad (\mathbf{k}_{m+1,n})_i = k(\mathbf{c}_{m+1}, \mathbf{x}_i), \quad (\mathbf{k}_{nc})_i = k(\mathbf{x}_i, \mathbf{c}_{m+1})$
$(\mathbf{k}_{mc})_i = k(\mathbf{c}_i, \mathbf{c}_{m+1}), \quad \mathbf{z} = {\mathbf{K}_\mathbf{c}}^{-1}\mathbf{k}_{mc}, \quad \eta = 1 - \mathbf{z}^T\mathbf{k}_{nc}, \quad (\mathbf{K}_{nm})_{ij} = k(\mathbf{x}_i, \mathbf{c}_j)$

SGMA es una técnica "Greedy" que selecciona en cada iteración un conjunto de candidatos e incorpora a la base el que obtiene un mayor descenso de error.

GPs

Modelo Completo:

- La solución de un GP se puede resolver analíticamente:
  $m(f(\mathbf{x}_i)) = \mathbf{k}_i^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$
  $v(f(\mathbf{x}_i)) = K(\mathbf{x}_i\mathbf{x}_i) - \mathbf{k}_i^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_i$

Modelo Semi-Paramétrico:

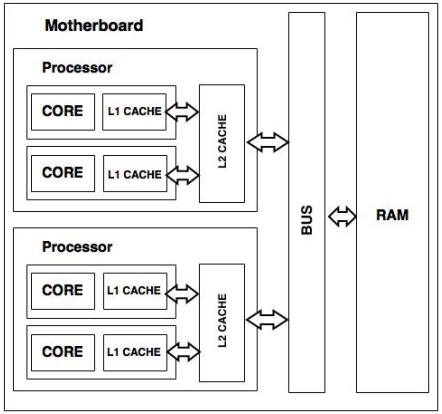
- Se puede hacer un modelo semi-paramétrico tomando un subconjunto de elementos base:
  $m(\mathbf{x}_i) = \mathbf{k}_i^T\mathbf{\Sigma}\mathbf{K}_{nm}\mathbf{y}$
  $v(\mathbf{x}_i) = \sigma^2 + \mathbf{k}_i^T\mathbf{\Sigma}\mathbf{k}_i$
  $\mathbf{\Sigma} = [\mathbf{K}_{nm}^T\mathbf{K}_{nm}/\sigma^2 + \mathbf{K}_{mm}]^{-1}$
- Para la obtención de los elementos se ha utilizado otra técnica Greedy, SGEV, que selecciona en cada iteración un conjunto de candidatos y añade el que maximiza la log-evidencia:
  $\mathcal{L} = \mathcal{Q} + \mathcal{G}$
  $\mathcal{Q} = \frac{1}{2\sigma^2}\mathbf{y}^T\mathbf{y} - \frac{1}{4\sigma^4}\mathbf{y}^T\mathbf{K}_{nm}\mathbf{\Sigma}\mathbf{K}_{nm}^T\mathbf{y}$
  $\mathcal{G} = \frac{1}{2}[log|\mathbf{\Sigma}| - log|\mathbf{K}_{mm}| + mlog\sigma^2]$

Factores influyentes en el rendimiento

Sistemas multinúcleo:

Tienen varias memorias caché que se organizan por niveles.

La memoria RAM es común a todos los procesos.



Factores de bajo nivel:

El acceso concurrente a recursos puede producir cuellos de botella.

Diferentes copias de los datos en distintas memorias caché.

Lectura de memoria más eficiente en datos consecutivos ya que disminuye el tiempo destinado al indexado.

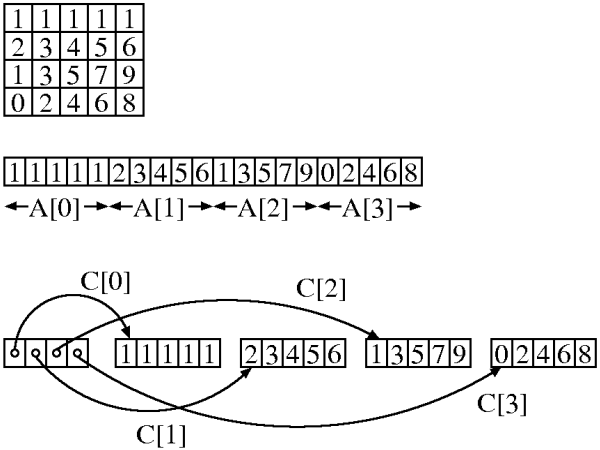
Factores de alto nivel:

Secciones de código no paralelizable $\rightarrow$ Cota máxima.

Tiempo de Comunicación entre Procesos.

Elecciones de diseño

Las matrices de funciones de núcleo y los conjuntos de entrenamiento se almacenan en bloques contiguos de memoria, de esta forma disminuye el tiempo de indexado:



Siempre que es posible cada núcleo trabaja sobre un área de memoria diferente.

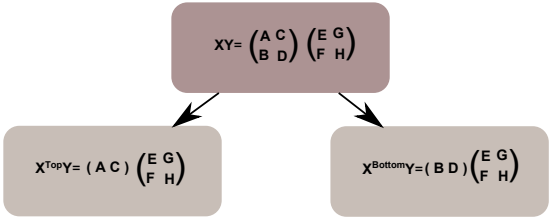
Memoria compartida $\rightarrow$ la comunicación entre procesos es instantánea.

El tiempo de ejecución de las partes no paralelas del código es absolutamente despreciable.

Operaciones

Producto de Matrices

Fácilmente paralelizable → La obtención de cada fila de la matriz resultado es independiente.



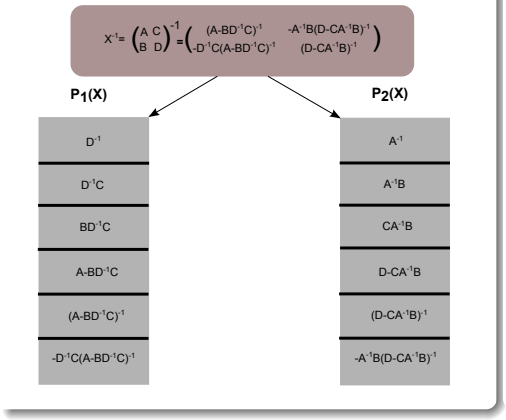
Inversión de matrices

Se hace uso de la inversión de una matriz por bloques.

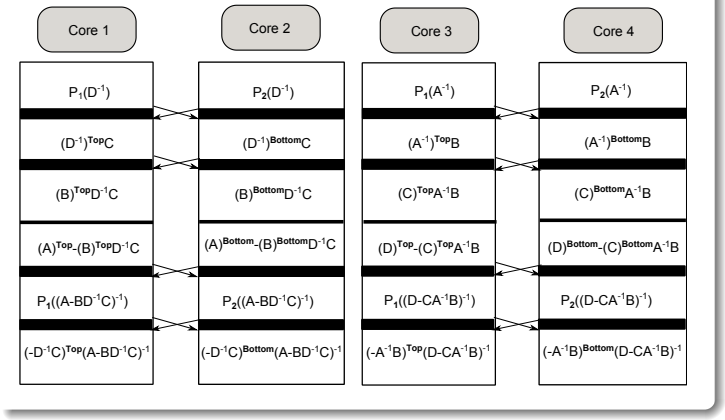
$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix}$$

Operaciones

Inversión con 2 núcleos



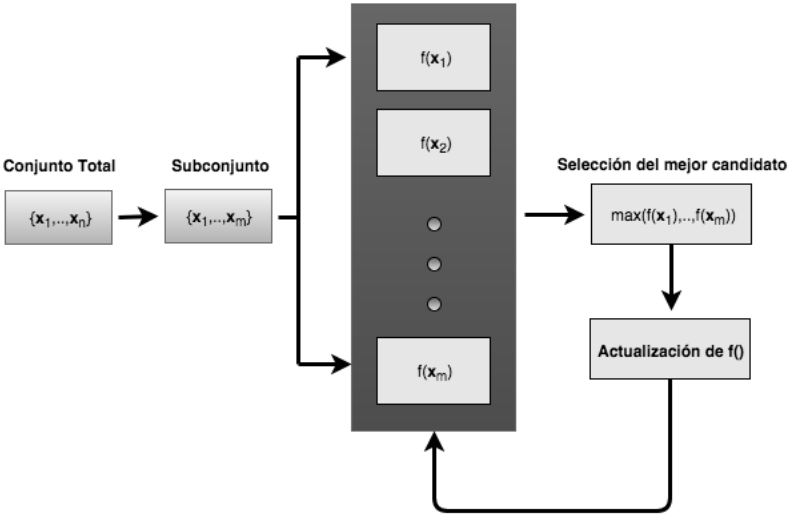
Inversión con 4 núcleos



Operaciones

Selección de los elementos base en modelos Semi-Paramétricos

Se ha utilizado el patrón de diseño Map Reduce



Experimentos

Algoritmos Implementados:

- **PS-SVM:** Implementación paralela de SVM Semi-Paramétrica.
- **P-GP:** Implementación paralela de GPs.
- **PS-GP:** Implementación paralela de GPs Semi-Paramétricos.

Detalles de Implmentación:

- Los algoritmos han sido implementados en C.
- Se ha utilizado OpenMP para implementar la paralización.
- Se ha utilizado la librería Intel MKL para las operaciones de álgebra lineal.

Benchmark:

- **LibSVM:** La librería más extendida de SVMs.
- **PSVM:** Implementación Dispersa de SVMs.
- **pyXGPR:** Librería de Procesos Gaussianos para regresión.

Experimentos

Servidor:

Característica	Valor
Modelo	HP DL160 G6
Frecuencia Base	3.06 GHz
Procesadores	2 Intel Xeon E5-2666 v3
Núcleos Reales	12 (2 x 6)
Memoria	48 GB
Caché	12 MiB
Hyperthreading	Si (24 Núcleos Virtuales, 12 Reales)
Sistema operativo	Linux Gentoo

Bases de datos SVMs:

- ADULT (32561/16281)
- WEB (24692/25057)
- USPS (7291/2007)
- MNIST (60000/10000)

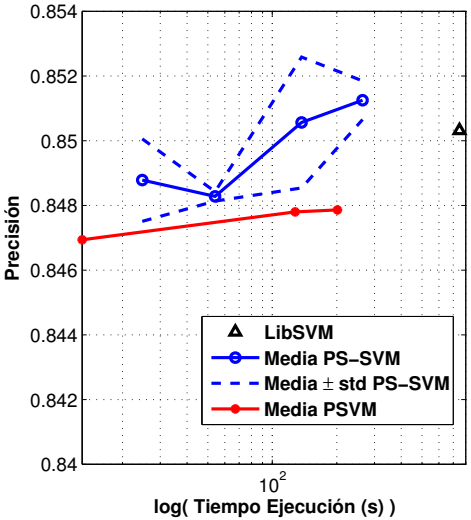
Bases de datos GPs:

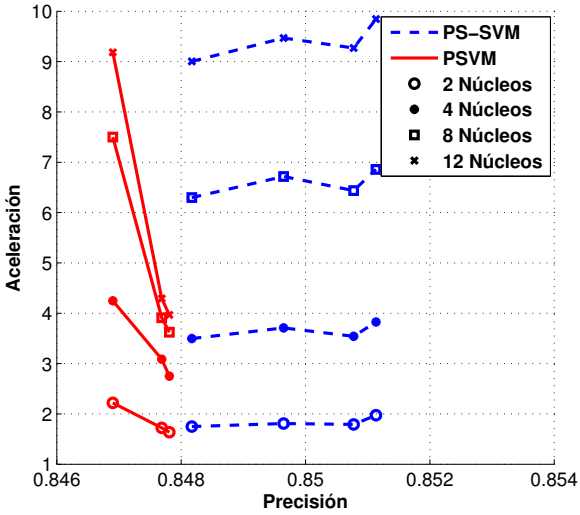
- Boston Housing (300/206)
- Abalone (3000/1177)
- Cadata (10000/10640)

Resultados SVM - Adult

Algoritmo	C	γ	Precisión Media(Desviación)	Tamaño
LibSVM	100000	0.0001	0.8503	10515
PS-SVM 50	100	0.0001	0.8488(1e-003)	50
PS-SVM 100	1000	0.1	0.8483(1e-004)	100
PS-SVM 200	1000	0.01	0.8506(2e-003)	200
PS-SVM 300	1	0.1	0.8512(6e-004)	300
PSVM 0.003	100	0.001	0.8469	11918
PSVM 0.008	10000	0.0001	0.8478	11526
PSVM 0.012	1	0.01	0.8479	11932

Resultados SVM - Adult

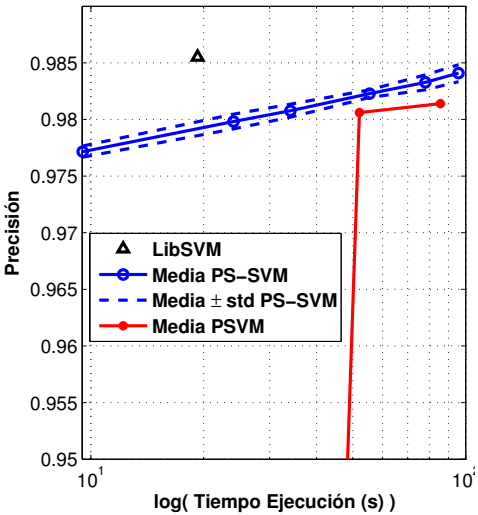


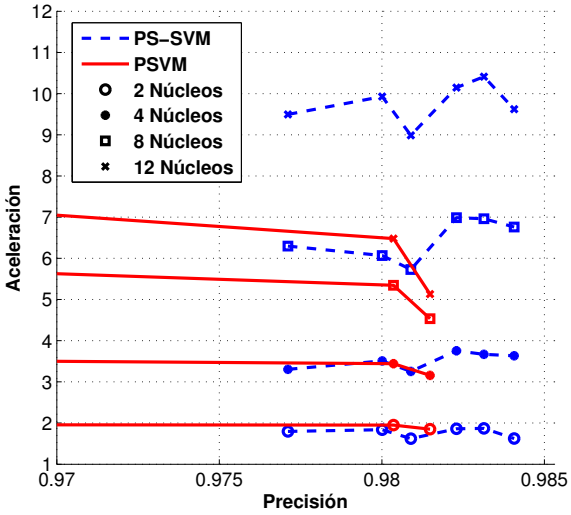


Resultados SVM - WEB

Algoritmo	C	γ	Precisión Media(Desviación)	Tamaño
LibSVM	100	0.1	0.9855	2531
PS-SVM 30	100	0.01	0.9772(5e-004)	30
PS-SVM 60	1000	0.1	0.9798(7e-004)	60
PS-SVM 90	1000	0.01	0.9808(6e-004)	90
PS-SVM 120	100	0.1	0.9823(3e-004)	120
PS-SVM 150	100	0.01	0.9833(7e-004)	150
PS-SVM 180	1000	0.01	0.9841(8e-004)	180
PSVM 0.002	1000	10	0.8944	24692
PSVM 0.004	1	0.01	0.8748	1533
PSVM 0.006	1	10	0.8944	24692
PSVM 0.008	100	0.0001	0.9806	1596
PSVM 0.010	100	0.001	0.9814	1398

Resultados SVM - WEB

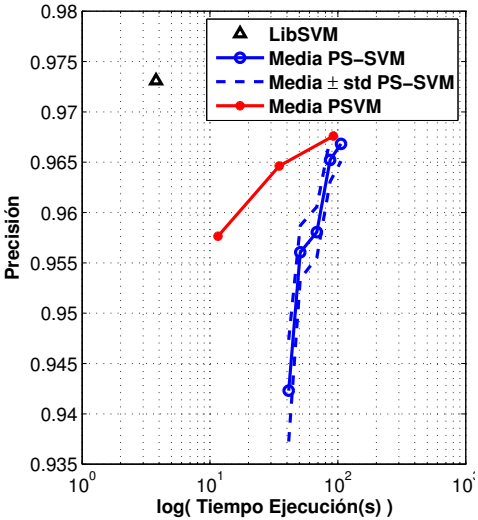


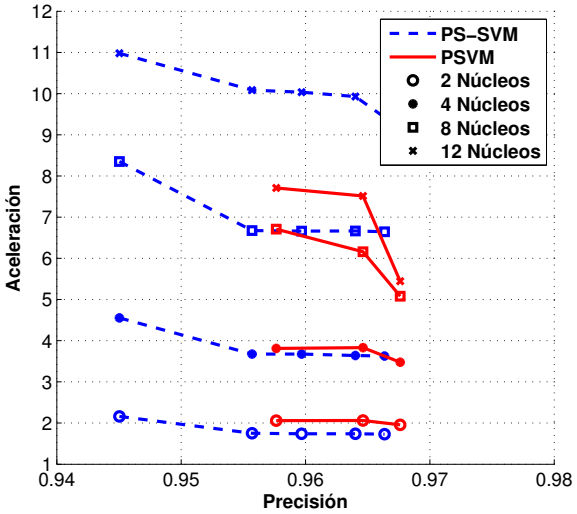


Resultados SVM - USPS

Algoritmo	C	γ	Precisión Media(Desviación)	Tamaño
LibSVM	100000	0.0001	0.9731	646
PS-SVM 60	1000	0.1	0.9423(5e-003)	60
PS-SVM 90	1000	0.001	0.9561(3e-003)	90
PS-SVM 120	1000	0.01	0.9581(3e-003)	120
PS-SVM 150	1000	0.001	0.9652(2e-003)	150
PS-SVM 180	1	0.1	0.9668(2e-003)	180
PSVM 0.004	100	0.001	0.9576	1088
PSVM 0.006	1000	0.0001	0.9646	678
PSVM 0.008	1000	0.0001	0.9676	623

Resultados SVM - USPS

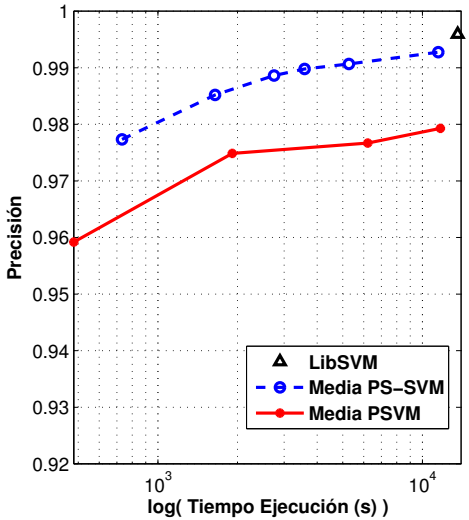


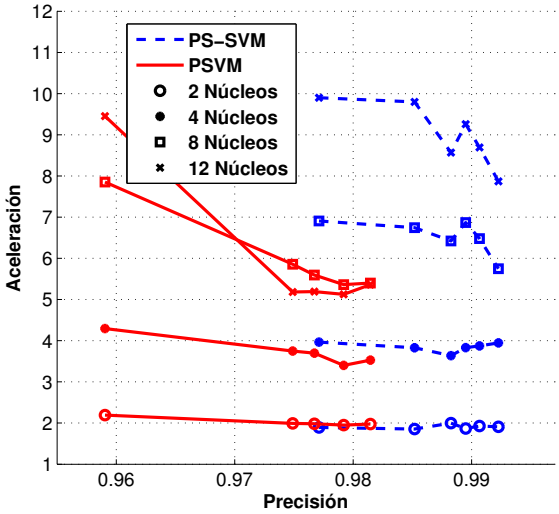


Resultados SVM - MNIST

Algoritmo	Precisión	Tamaño	Núcleos 1	2	4	8	12
LIBSVM	99.59	6850	13598	7273	3882	2369	2106
PS-SVM 100	97.73	100	665	352	168	96	67
PS-SVM 200	98.52	200	1496	806	391	222	153
PS-SVM 300	98.86	300	2500	1252	688	389	292
PS-SVM 400	98.98	400	3261	1747	851	475	352
PS-SVM 500	99.07	500	4795	2489	1239	740	552
PS-SVM 1000	99.28	1000	10446	5471	2646	1818	1327
PSVM 0.001	95.92	10498	481	219	112	61	51
PSVM 0.003	97.49	11708	1910	959	510	326	368
PSVM 0.005	97.67	11842	6215	3138	1680	1111	1197
PSVM 0.007	97.93	14429	11695	5992	3441	2182	2282
PSVM 0.009	98.14	11562	17631	8919	4998	3265	3289

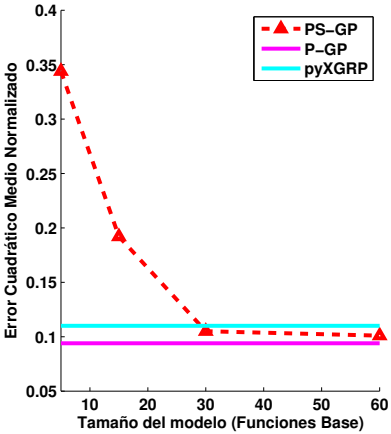
Resultados SVM - MNIST

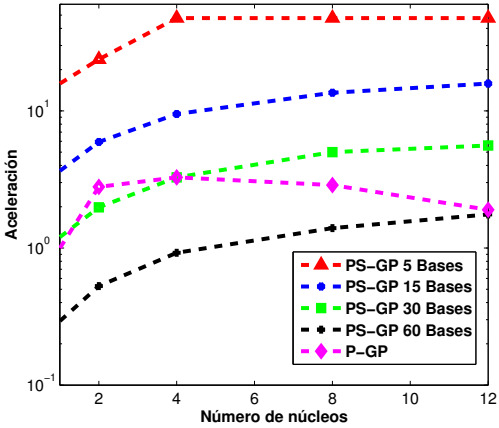




Resultados GPs - Boston Housing

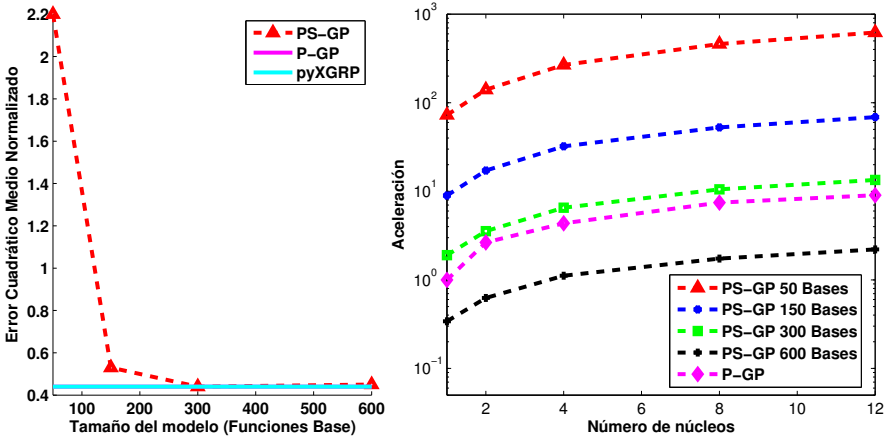
Algoritmo	NMSE
PS-GP$_{5\ \text{Bases}}$	0,344
PS-GP$_{15\ \text{Bases}}$	0,192
PS-GP$_{30\ \text{Bases}}$	**0,105**
PS-GP$_{60\ \text{Bases}}$	0,101
P-GP	**0,094**
pyXGPR	**0,11**





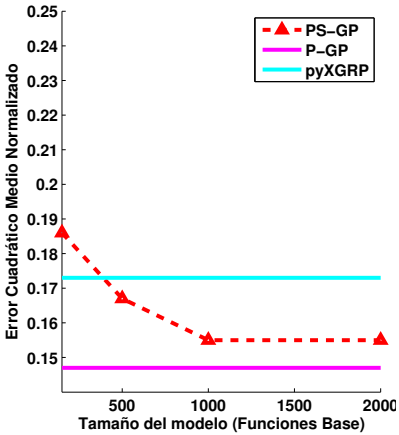
Resultados GPs - Abalone

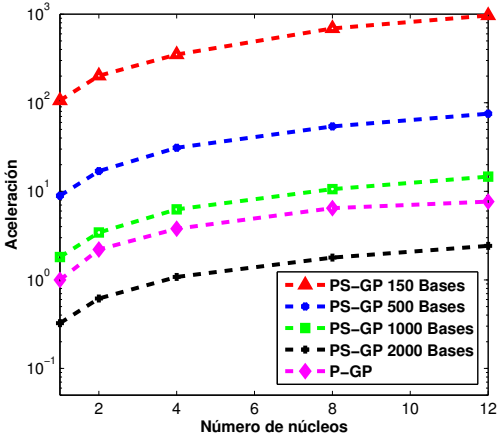
Algoritmo	NMSE
PS-GP$_{50\ \text{Bases}}$	$2{,}2$
PS-GP$_{150\ \text{Bases}}$	$0{,}53$
PS-GP$_{300\ \text{Bases}}$	$\mathbf{0{,}44}$
PS-GP$_{600\ \text{Bases}}$	$0{,}45$
P-GP	$\mathbf{0{,}44}$
pyXGPR	$\mathbf{0{,}44}$



Resultados GPs - Cadata

Algoritmo	NMSE
PS-GP$_{150\ \text{Bases}}$	0.186
PS-GP$_{500\ \text{Bases}}$	**0.167**
PS-GP$_{1000\ \text{Bases}}$	0.155
PS-GP$_{2000\ \text{Bases}}$	0.155
P-GP	**0.147**
pyXGPR	**0.173**





Conclusiones

- PS-SVM presenta una tasa de acierto igual o superior para el mismo tiempo de ejecución que PSVM.
- PS-SVM presenta una mejor aceleración en un rango mayor de valores de sus parámetros (número de elementos base) que PSVM (coeficiente de rango).
- PS-SVM produce modelos con un tamaño menor.
- Este tipo de técnicas puede ser empleadas en un amplio rango de algoritmos que trabajen de forma matricial.
- Combinar modelos semi-paramétricos con paralelización aumenta el campo de aplicación de los métodos de núcleo.

Posibles mejoras respecto al anterior modelo

Limitaciones del modelo anterior

- A la hora de resolver un sistema de ecuaciones existen soluciones más eficientes que obtener la inversa de una matriz.
- El cómputo de la inversa por bloques puede representar cierta inestabilidad numérica.

Alternativa

- Las matrices con las que trabajamos son definidas positivas.
- La factorización de Cholesky permite resolver sistemas de ecuaciones cuando la matriz del sistema es definida positiva de forma muy eficiente.

Operaciones - Producto Matricial

División QuadTree del producto de matrices:

$$\begin{bmatrix} \mathbf{A_1} & \mathbf{A_2} \\ \mathbf{A_3} & \mathbf{A_4} \end{bmatrix} = \begin{bmatrix} \mathbf{B_1} & \mathbf{B_2} \\ \mathbf{B_3} & \mathbf{B_4} \end{bmatrix} \begin{bmatrix} \mathbf{C_1} & \mathbf{C_2} \\ \mathbf{C_3} & \mathbf{C_4} \end{bmatrix}$$

Subtareas:

$$\text{Subtarea 1} \begin{cases} \mathbf{A_1} = \mathbf{B_1}\mathbf{C_1} + \mathbf{B_2}\mathbf{C_3} \\ \mathbf{A_2} = \mathbf{B_1}\mathbf{C_2} + \mathbf{B_2}\mathbf{C_4} \end{cases}$$

$$\text{Subtarea 2} \begin{cases} \mathbf{A_3} = \mathbf{B_3}\mathbf{C_1} + \mathbf{B_4}\mathbf{C_3} \\ \mathbf{A_4} = \mathbf{B_3}\mathbf{C_2} + \mathbf{B_4}\mathbf{C_4} \end{cases}$$

Operaciones - Sumas y Restas Matriciales

En el caso de sumas y restas Matriciales:

$$\begin{bmatrix} \mathbf{A_1} & \mathbf{A_2} \\ \mathbf{A_3} & \mathbf{A_4} \end{bmatrix} = \begin{bmatrix} \mathbf{B_1} & \mathbf{B_2} \\ \mathbf{B_3} & \mathbf{B_4} \end{bmatrix} + \begin{bmatrix} \mathbf{C_1} & \mathbf{C_2} \\ \mathbf{C_3} & \mathbf{C_4} \end{bmatrix}$$

Subtareas:

$$\text{Subtarea 1} \begin{cases} \mathbf{A_1} = \mathbf{B_1} + \mathbf{C_1} \\ \mathbf{A_2} = \mathbf{B_2} + \mathbf{C_2} \end{cases}$$

$$\text{Subtarea 2} \begin{cases} \mathbf{A_3} = \mathbf{B_3} + \mathbf{C_3} \\ \mathbf{A_4} = \mathbf{B_4} + \mathbf{C_4} \end{cases}$$

Operaciones - Inversión Triangular

La inversión triangular se simplifica enormemente respecto a una inversión completa:

$$\begin{bmatrix} \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} \end{bmatrix} = \begin{bmatrix} \mathbf{L_1} & \mathbf{0} \\ \mathbf{L_3} & \mathbf{L_4} \end{bmatrix} \begin{bmatrix} (\mathbf{L^{-1}})_\mathbf{1} & \mathbf{0} \\ (\mathbf{L^{-1}})_\mathbf{3} & (\mathbf{L^{-1}})_\mathbf{4} \end{bmatrix}$$

Subtareas:

Subtarea 1 $\left\{ (\mathbf{L^{-1}})_\mathbf{1} = \mathbf{L_1}^{-1} \right.$

Subtarea 2 $\left\{ (\mathbf{L^{-1}})_\mathbf{4} = \mathbf{L_4}^{-1} \right.$

$(\mathbf{L^{-1}})_\mathbf{3} = -(\mathbf{L^{-1}})_\mathbf{4}\mathbf{L_3}(\mathbf{L^{-1}})_\mathbf{1} \Rightarrow$ Paralelizable

Operaciones - Factorización de Cholesky

La factorización de Cholesky descompone una matriz en una matriz triangular inferior por su transpuesta conjugada:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{L_1} & \mathbf{L_2} \\ \mathbf{L_3} & \mathbf{L_4} \end{bmatrix} \begin{bmatrix} \mathbf{L_1^*} & \mathbf{L_3^*} \\ \mathbf{L_2^*} & \mathbf{L_4^*} \end{bmatrix}$$

$$\mathbf{L_2} = \mathbf{0}$$
$$\mathbf{A} = \mathbf{L_1}\mathbf{L_1^*} \Rightarrow \mathbf{L_1} = Chol(\mathbf{A})$$
$$\mathbf{C} = \mathbf{L_3}\mathbf{L_1^*} \Rightarrow \mathbf{L_3} = \mathbf{C}\mathbf{L_1^*}^{-1}$$
$$\mathbf{D} = \mathbf{L_3}\mathbf{L_3^*} + \mathbf{L_4}\mathbf{L_4^*} \Rightarrow \mathbf{L_4} = Chol(\mathbf{D} - \mathbf{L_3}\mathbf{L_3}^*)$$

Experimentos

Algoritmos Implementados:

- **PIRWLS:** Implementación paralela de SVMs.
- **PSIRWLS:** Implementación paralela de SVMs Semi-Paramétricas.

Se ha liberado como proyecto de software libre:

- **PIRWLS:** https://github.com/RobeDM/PIRWLS.git
- **PSIRWLS:** https://github.com/RobeDM/PSIRWLS.git

Detalles de Implmentación:

- Los algoritmos han sido implementados en C.
- Se ha utilizado OpenMP para implementar la paralización.
- Se ha utilizado la librería Lapack para las operaciones de álgebra lineal.

Benchmark:

- **LibSVM:** La librería más extendida de SVMs.
- **PS-SVM:** Implementación Dispersa de SVMs.

Experimentos

Servidor:

Característica	Valor
Modelo	c4.8xlarge
Instrucciones	64-bit
Frecuencia Base	2.9 GHz
Procesadores	2 Intel Xeon E5-2666 v3
Núcleos Reales	18 (2 x 9)
Memoria	60 MB
Caché	25 MiB
Hyperthreading	Si (36 Núcleos Virtuales, pero 18 Reales)
Sistema Operativo	Linux Ubuntu

Bases de datos:

- Splice (2175 / 1000)
- Adult (32561 / 16281)
- MNIST (60000 / 10000)
- CovType (522910 / 58102)

Resultados

Cuadro: Resultados de LibSVM

	Precisión	Tiempo (s)	Tamaño
SPLICE	89 %	0,37	1053
ADULT	85,1 %	88	11429
MNIST	98,9 %	1008	5376
COVTYPE	92,5 %	18499	306143

Cuadro: Resultados de PIRWLS

	Precisión	Tiempo (s)			Tamaño
		1 Núcleo	8 Núcleos	16 Núcleos	
SPLICE	88,8 %	7,6	1,7	1,4	1090
ADULT	85,1 %	287	41,6	27,0	11431
MNIST	98,9 %	2373	329	168	6048
COVTYPE	92,5 %	33763	4872	2751	305663

Resultados

Cuadro: PS-SVM (50 Centroides)

	Precisión	Tiempo (s)			Tamaño
		1 Núcleo	8 Núcleos	16 Núcleos	
SPLICE	83,4 %	0,264	0,237	0,24	50
ADULT	85,1 %	18,3	3,3	1,9	50
MNIST	90,7 %	235	34,8	22,1	50
COVTYPE	75,7 %	357	61	43	50

Cuadro: PS-SVM (500 Centroides)

	Precisión	Tiempo (s)			Tamaño
		1 Núcleo	8 Núcleos	16 Núcleos	
SPLICE	88,4 %	30,1	4,6	4,1	500
ADULT	85,1 %	750	131	111	500
MNIST	97,43 %	2778	526	351	500
COVTYPE	80,8 %	11028	2070	1540	500

Resultados

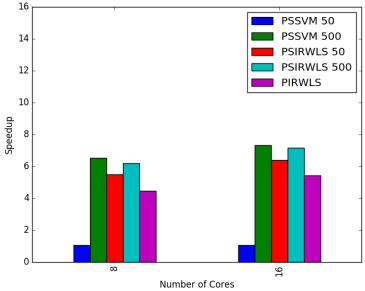
Cuadro: PSIRWLS (50 Centroides)

	Precisión	Tiempo (s)			Tamaño
		1 Núcleo	8 Núcleos	16 Núcleos	
SPLICE	83,4 %	0,77	0,14	0,12	50
ADULT	85,1 %	11,2	1,7	1,2	50
MNIST	90,7 %	139	19,6	10,2	50
COVTYPE	75,7 %	147	20	12,3	50

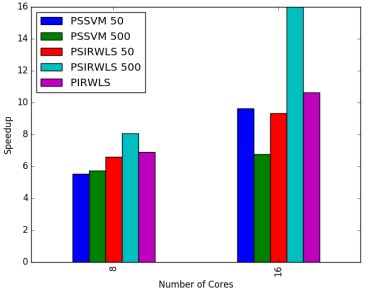
Cuadro: PSIRWLS (500 Centroides)

	Precisión	Tiempo (s)			Tamaño
		1 Núcleo	8 Núcleos	16 Núcleos	
SPLICE	88,4 %	22,9	3,7	3,2	500
ADULT	85,1 %	589	72,9	36,4	500
MNIST	97,43 %	2243	322	169	500
COVTYPE	80,8 %	9009	1101	575	500

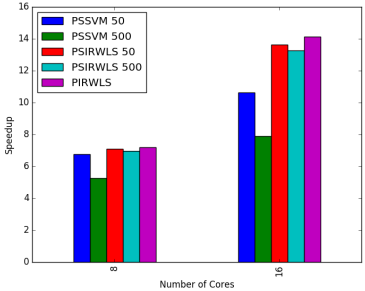
Resultados



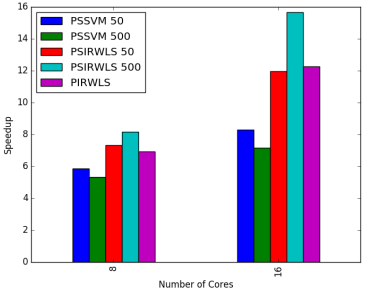
(a) SPLICE



(b) ADULT



(c) MNIST

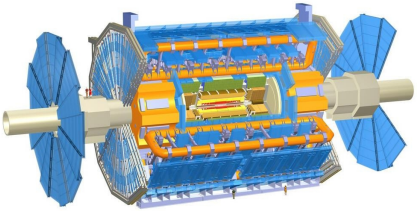


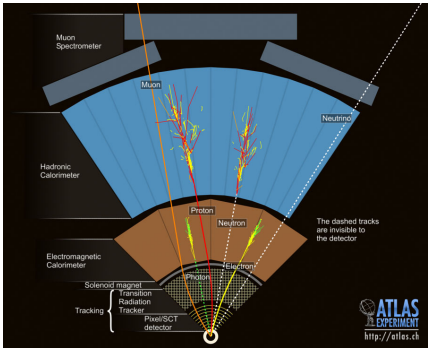
(d) COVTYPE

Bosón de Higgs

- Conjunto de datos:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1, w_1), ..., (\mathbf{x}_n, y_n, w_n)\},$$
$$\mathbf{x}_i \in \mathbb{R}^d,$$
$$y_i \in \{r, s\},$$
$$w_i \in \mathbb{R}^+.$$





- Función de evaluación:

$$AMS = \sqrt{2\left((s_T + r_T + r_r)log\left(1 + \frac{s_T}{r_T + r_r}\right) - s_T\right)},$$

$$s_T = \sum_{i=1}^{n} w_i 1\{y_i = s\}1\{\hat{y}_i = s\},$$

$$r_T = \sum_{i=1}^{n} w_i 1\{y_i = r\}1\{\hat{y}_i = s\}.$$

Bosón de Higgs

Preprocesado

- Preprocesado paralelo
- Transformación del espacio de entrada
- Imputación de valores desconocidos

Algoritmos

- Se seleccionaron algoritmos que tuviesen implementación paralela (PIRWLS, Perceptrón Multicapa, Random Forest, Gradient Boosted Trees).
- El criterio de entrenamiento fue maximizar el WAUC.

$$WTPR = \frac{\sum_{i=1}^{n} w_i 1\{y_i = s\} 1\{\hat{y}_i = s\}}{\sum_{i=1}^{n} w_i 1\{y_i = s\}},$$

$$WFPR = \frac{\sum_{i=1}^{n} w_i 1\{y_i = r\} 1\{\hat{y}_i = s\}}{\sum_{i=1}^{n} w_i 1\{y_i = r\}}.$$

Esquema Final

- Varios conjuntos de entrenamiento con expansiones polinómicas.
- Entrenamiento multinúcleo con memoria compartida.
- Estabilización con Bagging.
- Ensemble fusionando salidas mediante combinación lineal.

Conclusiones

- Conjunto de test con 550000 eventos
- Clasificación pública 18 %, privada 82 %
- AMS = 3.76
- 9º de 1785 equipos.

Rastreo entre Dispositivos

- Conjunto de datos:
  - Base de datos relacional con información sobre dispositivos, cookies, direcciones IP y comportamiento.
  - Información de alto nivel sobre dispositivos y cookies.
  - Información de actuación de estos dispositivos y cookies y en direcciones IP.
  - Información sobre las direcciones IP y aplicaciones móviles.
- Función de evaluación: El objetivo es determinar que cookies pertenecen a un individuo utilizando un dispositivo. La métrica de evaluación es el $\mathcal{F}_{0,5}$.

$$\mathcal{F}_\beta = (1+\beta^2)\frac{pr}{\beta^2 p + r},$$
$$p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}$$

Rastreo entre Dispositivos

Preprocesado

- Preprocesado paralelo
- Se limite el conjunto de cookies candidatas de cada dispositivo con reglas basadas en las direcciones IP que ambos compartían.

Conjunto de entrenamiento

- Cada muestra representa una pareja dispositivo/cookie candidata.
- Contiene características del dispositivo, de la cookie y de cómo se relacionan en el grafo.

Algoritmos

- Se seleccionaron algoritmos que tuviesen implementación paralela.
- Estabilización con Bagging.

Aprendizaje Semisupervisado

- Cuando sólo un candidato obtiene una probabilidad alta es extremadamente probable que esas sean sus etiquetas verdaderas.
- El modelo se volvió a entrenar utilizando estas muestras en el conjunto de entrenamiento.

Conclusiones

- Conjunto de test con 61156 dispositivos.
- Clasificación pública 30 %, privada 70 %
- $\mathcal{F}_{0,5} = 0{,}88$
- Finalizó 3º de 340 equipos.

Conclusiones

- Se han propuesto nuevos esquemas de paralización que pretender reducir las limitaciones de los métodos de núcleo.
- Se han tomado criterios de diseño de alto y bajo nivel para conseguir buenas prestaciones de aceleración.
- Se ha propuesto un esquema que hace uso de la inversión matricula por bloques (PS-SVM, P-GP, PS-GP).
- Se ha propuesto un nuevo esquema que hace uso de una Factorización de Cholesky paralela (PIRWLS, PSIRWLS).
- El código fuente está disponible como software libre.
- Se ha demostrado la utilidad práctica de la computación paralela en competiciones de aprendizaje automático.

Trabajo Futuro

- Extensión de la formulación para otros tipos de algoritmos.
- Adaptación de los esquemas para su utilización en GPUs.
- Extender el trabajo para entornos distribuidos.
- Implementación propiamente “Big Data”.

Contribuciones

Principales

- Díaz-Morales, R., Molina-Bulla, H. Y., and Navia-Vázquez, A. (2011). Parallel Semiparametric Support Vector Machines. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 475-481. IEEE.
- Díaz-Morales, R., and Navia-Vázquez, A. (forthcomming, In press). Efficient Parallel Implementation of Kernel Methods. In *Neurocomputing*. Elsevier.
- Díaz-Morales, R., and Navia-Vázquez, A. (Under Review). Improving the efficiency of IRWLS SVCs using Parallel Cholesky Factorization. In *Pattern Recognition Letters*. Elsevier.

Secundarias

- Díaz-Morales, R. and Navia-Vázquez, A. (2015). Optimization of AMS using Weighted AUC optimized models. In *JMLR W&CP, 42*, pages 109-127.
- Díaz-Morales, R. (2015). Cross-Device Tracking: Matching Devices and Cookies. In *Data Mining Workshops (ICDMW), 2015 IEEE International Conference on*, pages 1699-1704. IEEE

Preguntas