ITAM

Estudio numérico de algoritmos a gran escala para MSV

Pablo Campos Viana

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Junio 2016

Índice

Motivación

- El método Máquinas de Soporte Vectorial (**MSV**) es ampliamente utilizado para problemas de clasificación binaria.
- La gran mayoría de métodos utilizados en el aprendizaje estadístico se implementan con técnicas que divergen de las utilizadas en los métodos clásicos de la optimización numérica.
- Implementaciones del método MSV:
  - Variantes de *programación cuadrática* en la formulación dual. Ej. **LIBSVM**.
  - Variantes del método *descenso del gradiente estocástico* en la formulación primal. Ej. **SVM-SGD**.

- **Aprendizaje supervisado**: $\mathbf{z} = (\mathbf{x}, y)$ en donde $\mathbf{x} \in \mathcal{X}$ y $y \in \mathcal{Y}$, en donde $\mathcal{X}$ es el espacio de características y $\mathcal{Y}$ el espacio de posibles respuestas.
- Muestra disponible:
  $\mathcal{S} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$.
- Se asume que existe una distribución de probabilidad desconocida sobre el espacio $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, es decir, que existe una distribución de probabilidad $P(\mathbf{z}) = P(\mathbf{x}, y)$.
- Problema: Encontrar una función $f \in \mathcal{F}, f : \mathcal{X} \to \mathcal{Y}$ tal que $f(\mathbf{x}) \approx y$.
- *Funcional de pérdida*: $\ell(f(\mathbf{x}), y)$.

Aprendizaje estadístico

- *Riesgo esperado*:

$$E(f) = \int_{\mathcal{Z}} \ell(f(\mathbf{x}), y) dP(\mathbf{z})$$

- *Riesgo empírico*:

$$E_{\mathcal{S}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)$$

- **Regularización**: Si la función $f \in \mathcal{F}$ está parametrizada por un vector $\mathbf{w}$ entonces la función a minimizar es:

$$Q(\mathbf{w}) = E_{\mathcal{S}}(f_{\mathbf{w}}) + \lambda R(\mathbf{w})$$

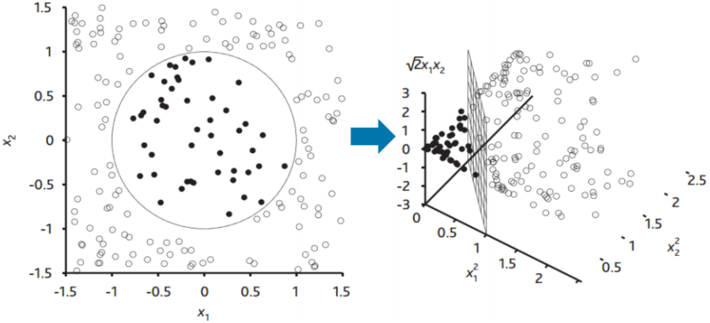
Aprendizaje de pequeña y gran escala

- *Pequeña escala*: Restringidos por el tamaño del conjunto de datos.
- *Gran escala*: Restringidos por el tiempo de cómputo. Los algoritmos de optimización pueden lograr una mejor capacidad de generalización.

Transformación explícita de las observaciones

- En ocasiones, el uso de cierta clase de métodos no es apropiado para construir una función de predicción para un conjunto de datos dado.
- Realizar una transformación de las variables puede resultar de gran ayuda para la construcción de una función de predicción.
- Función de transformación: $\Phi : \mathcal{X} \rightarrow \mathcal{H}$.
- Muestra de entrenamiento: $\mathcal{S} = \{\Phi(\mathbf{x}_i), y_i\}_{i=1}^{n}\,, \quad \Phi(\mathbf{x}_i) \in \mathcal{H}, y_i \in \mathcal{Y}$.

Transformación explícita: Ejemplo



- Transformación: $\Phi(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2)$.

Transformación implícita a través de Kernels

- Una función Kernel K es una función $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ que es continua, simétrica y positiva definida.
- El Kernel realiza una evaluación entre cualquier pareja de observaciones $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$.
- Cuando se considera una muestra $\mathcal{S}$, el Kernel finito puede ser representado como una matriz en donde $K_{\mathcal{S}}(\mathbf{x}_i, \mathbf{x}_j) = k_{i,j}$.

Transformación implícita a través de Kernels

Teorema de Moore-Aronszajn (1950)

Dado un Kernel K, existe un único espacio $\mathcal{H}$ tal que $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$

Ejemplos:

- Polinomial no homogéneo: $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^{\intercal}\mathbf{x}_j + c)^d$, para $c \in \mathbb{R}$ y $d \in \mathbb{N}$.
- Tangente hiperbólica: $k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i^{\intercal}\mathbf{x}_j + c)$, para $\kappa > 0$ y $c < 0$.
- Gaussiano de base radial (RBF): $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, para $\gamma > 0$.

Formulación del método MSV

- Dadas dos clases $y \in \{-1, 1\}$, una nueva observación $\boldsymbol{x}$ puede ser clasificada de acuerdo a una función $f(\boldsymbol{x}) = \mathbf{w}^\intercal \boldsymbol{x} + b$, o bien, $f(\boldsymbol{x}) = \mathbf{w}^\intercal \Phi(\boldsymbol{x}) + b$, mediante una función Φ.
- El hiperplano obtenido mediante $f(\boldsymbol{x}) = 0$ define una frontera de decisión en el espacio transformado y los parámetros $\mathbf{w}$ y b son determinados a partir de la muestra disponible de los datos.
- La formulación método MSV contiene tres ingredientes escenciales:
  - Hiperplanos óptimos.
  - Margen suave.
  - La técnica del Kernel.

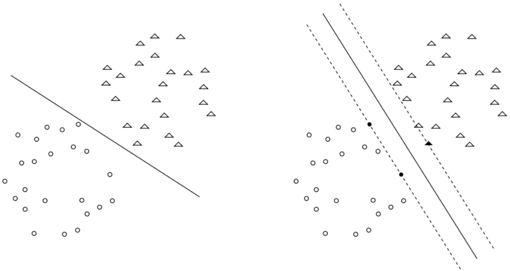
Hiperplanos óptimos

- Dadas dos clases $y \in \{-1, 1\}$, una nueva observación $\boldsymbol{x}$ puede ser clasificada de acuerdo a una función $f(\boldsymbol{x}) = \mathbf{w}^\intercal \boldsymbol{x} + b$, o bien, $f(\boldsymbol{x}) = \mathbf{w}^\intercal \Phi(\boldsymbol{x}) + b$, mediante una función Φ.
- El margen representa la distancia del hiperplano hacia el punto más cercano de cada una de las dos clases.
- El problema puede ser formulado de la siguiente forma.

$$\min_{\mathbf{w},b} \quad \mathcal{P}(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{sujeto a:} \quad y_i\left(\mathbf{w}^\intercal \Phi(\mathbf{x}_i) + b\right) \geq 1, \quad \forall i$$

Hiperplanos óptimos: Ejemplo



(a) (b)

- El plano separador de margen máximo se define mediante la ecuación $f(\mathbf{x}) = \mathbf{w}^\intercal \Phi(\mathbf{x}) + b = 0$ a través del vector $\mathbf{w}$ que resuelve el problema primal.

Hiperplanos óptimos

- Utilizando Teoría de Dualidad, se puede establecer el problema equivalente.

$$\max_{\boldsymbol{\alpha}} \quad \mathcal{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^\intercal \Phi(\mathbf{x}_j)$$

$$\text{sujeto a:} \quad \alpha_i \geq 0 \quad \forall i$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \forall i$$

- Solución: $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(\mathbf{x}_i)$.
- Función de decisión óptima:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(\mathbf{x}_i)^\intercal \Phi(\boldsymbol{x}) + b^*$$

Margen suave

- Cuando el problema no es linealmente separable, se pueden añadir variables de holgura.
- Problema primal:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \mathcal{P}(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{sujeto a:} \quad y_i\left(\mathbf{w}^\intercal\Phi(\mathbf{x}_i)+b\right) \geq 1-\xi_i, \quad \forall i$$

$$\xi_i \geq 0, \quad \forall i$$

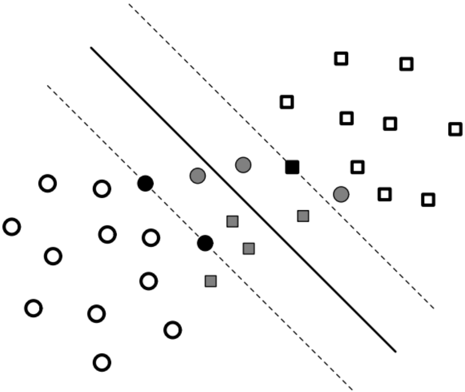
- Problema dual:

$$\max_{\boldsymbol{\alpha}} \quad \mathcal{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}^{n}\alpha_i\alpha_j y_i y_j \Phi(\mathbf{x}_i)^\intercal\Phi(\mathbf{x}_j)$$

$$\text{sujeto a:} \quad 0 \leq \alpha_i \leq C \quad \forall i$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \quad \forall i$$

Margen suave



- El problema puede ser resuelto al permitir que algunos de los datos violen las restricciones del margen.

La técnica del Kernel

- Reemplazar el producto interno por evaluaciones mediante un Kernel.
- Problema dual utilizando un Kernel:

$$\max_{\boldsymbol{\alpha}} \quad \mathcal{D}(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j K_{ij}$$

$$\text{sujeto a:} \quad 0 \leq \alpha_i \leq C \quad \forall i$$
$$\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \forall i$$

- Solución:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \boldsymbol{x}) + b^*$$

Problemas prácticos del método

- **Calcular valores Kernel es costoso.** Calcular cada valor de Kernel involucra la manipulación de volúmenes de datos que pueden ser muy grandes.
- **Calcular la matriz Kernel completa es ineficiente.** El tiempo total de entrenamiento es usualmente más bajo que el tiempo requerido para calcular la matriz Kernel completa.
- **La matriz Kernel no cabe en memoria.** Cuando el número de datos de entrenamiento crece, la matriz Kernel se vuelve muy grande y no puede mantenerse guardada en memoria.

El método de descomposición

- Los métodos de descomposición intentan resolver el problema dual mediante la resolución de una secuencia de subproblemas cuadráticos más simples.
- En cada iteración, se optimiza un subconjunto de coeficientes $\alpha_i, i \in \mathcal{B}$ y mantiene el resto de coeficientes $\alpha_j, j \notin \mathcal{B}$ sin modificar.
- Subproblema:

$$\max_{\boldsymbol{\alpha}'} \quad \mathcal{D}(\boldsymbol{\alpha}') = \sum_{i=1}^{n} \alpha_i' - \frac{1}{2} \sum_{i,j}^{n} \alpha_i' \alpha_j' y_i y_j K_{ij}$$

$$\text{sujeto a:} \quad 0 \leq \alpha_i' \leq C \quad \forall i \in \mathcal{B}$$
$$\alpha_i' = \alpha_i \quad \forall i \notin \mathcal{B}$$
$$\sum_{i=1}^{n} \alpha_i' y_i = 0 \quad \forall i$$

Optimización mínima secuencial

- Cada subproblema sucesivo de programación cuadrática tiene solamente dos variables.
- La restricción de igualdad hace a cada subproblema uno de optimización de una sola dimensión.
- Procedimiento:
    - Elegir un coeficiente α_i que viole la condición de optimalidad del problema.
    - Elegir un coeficiente α_j y optimizar el subproblema para la pareja α_i, α_j.
    - Repetir hasta convergencia.

Elección del conjunto activo

- Las restricciones $0 \leq \alpha_i \leq C$ pueden ser expresadas en función de las variables $\alpha_i y_i$ de la siguiente forma:

$$y_i \alpha_i \in [A_i, B_i] = \begin{cases} [0, +C] & \text{si } y_i = +1 \\ [-C, 0] & \text{si } y_i = -1 \end{cases}$$

- Solamente dos coeficientes de la solución $\boldsymbol{\alpha'}$ del algoritmo OMS difieren de los coeficientes del vector $\boldsymbol{\alpha}$, por lo cual $\boldsymbol{\alpha'} = \boldsymbol{\alpha} + \lambda \mathbf{u}$.
- Es suficiente considerar direcciones $\mathbf{u}^{ij} = (u_1^{ij}, \ldots, u_n^{ij})$ tales que:

$$u_k^{ij} = \begin{cases} +y_i & \text{si } k = i \\ -y_j & \text{si } k = j \\ 0 & \text{en otro caso} \end{cases}$$

Elección del conjunto activo

- Elección por máxima ganancia:

$$\mathbf{u}^* = \text{arg} \max_{\mathbf{u}^{ij} \in \mathcal{U}} \max_{0 \leq \lambda} \mathcal{D}(\boldsymbol{\alpha} + \lambda \mathbf{u}^{ij}) - \mathcal{D}(\boldsymbol{\alpha})$$
$$\text{sujeto a} \quad y_i \alpha_i + \lambda \leq B_i, \quad y_j \alpha_j - \lambda \geq A_j$$

- Elección por pareja de máxima violación:

$$\mathcal{D}(\boldsymbol{\alpha} + \lambda \mathbf{u}^{ij}) - \mathcal{D}(\boldsymbol{\alpha}) \approx \lambda \mathbf{g}^{\mathsf{T}} \mathbf{u}^{ij}$$

$$\mathbf{u}^* = \text{arg} \max_{\mathbf{u}^{ij} \in \mathcal{U}} \mathbf{g}^{\mathsf{T}} \mathbf{u}^{ij}$$

Elección del conjunto activo

- Elección por criterio de segundo orden:

$$\mathcal{D}(\boldsymbol{\alpha}+\lambda\mathbf{u}^{ij})-\mathcal{D}(\boldsymbol{\alpha}) \approx \lambda(y_i g_i - y_j g_j) - \frac{\lambda^2}{2}(K_{ii}+K_{jj}-2K_{ij})$$

- El cálculo para el índice i es exacto de acuerdo al esquema de máxima pareja de violación.
- El cálculo para el índice j puede ser realizado en un tiempo proporcional a n.
- Se requiere la diagonal de la matriz Kernel, la cual es almacenada en memoria caché.

La estrategia de *reducción*

- Se reduce el tamaño del problema al eliminar temporalmente algunas variables α_i que son improbables de ser seleccionadas como parte del conjunto activo.
- Partición de los datos de entrenamiento:
  - Datos que no son vectores de soporte $(\alpha_i = 0)$.
  - Vectores de soporte acotados $(\alpha_i = C)$.
  - Vectores de soporte libres.
- La rutina de *reducción* es invocada cada $\min(n, 1000)$ iteraciones.
- Dado que esta estregia es relativamente agresiva, una rutina para revertir la *reducción* es invocada periódicamente.

Descenso del gradiente

- Descenso del gradiente para encontrar el mínimo de una función $f(\mathbf{w})$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t)$$

- En el contexto de los problemas de aprendizaje:

$$Q(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i; \mathbf{w}) + \lambda R(\mathbf{w})$$

$$\nabla Q(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell(\mathbf{x}_i, y_i; \mathbf{w}) + \lambda \nabla R(\mathbf{w})$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left[\frac{1}{n}\sum_{i=1}^{n} \nabla \ell(\mathbf{x}_i, y_i; \mathbf{w}_t) + \lambda \nabla R(\mathbf{w}_t)\right]$$

Descenso del gradiente en MSV

- El problema primal del método MSV puede ser escrito de la siguiente forma:

$$\frac{1}{n}\sum_{i=1}^{n}[1 - f(\mathbf{x}_i;\mathbf{w})y_i]_+ + \lambda\|\mathbf{w}\|^2$$

- La regla de actualización toma la forma:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \begin{cases} \lambda\mathbf{w}_t & \text{si } f(\mathbf{x}_t;\mathbf{w}_t)y_t > 1 \\ \lambda\mathbf{w}_t - y_t\mathbf{x}_t & \text{en otro caso} \end{cases}$$

Descenso del gradiente estocástico

- Una modificación simple es calcular el gradiente respecto a una sola observación.
- Bajo este enfoque, la regla del método para $Q(\mathbf{w})$ está dada por:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left[\nabla \ell(\mathbf{x}_{i(t)}, y_{i(t)}; \mathbf{w}_t) + \lambda \nabla R(\mathbf{w}_t) \right]$$

- Tasa de aprendizaje (Shalev-Shwartz et al., 2007):

$$\eta_t = \frac{1}{\lambda(t_0 + t)}$$

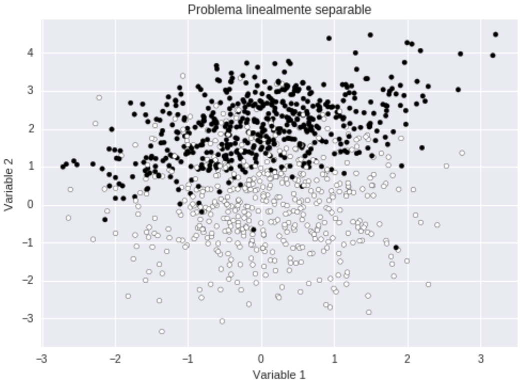
Experimentos numéricos

- **Problemas simulados**: Variando el tamaño del conjunto de datos de entrenamiento.
- **Problemas reales**: Conjuntos de datos inspirados en problemas reales, obtenidos de repositorios populares.

Problemas simulados: Parámetros

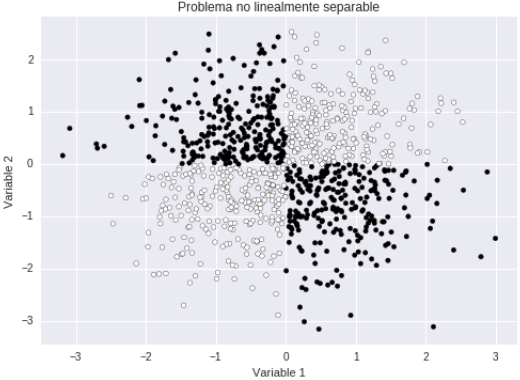
- Tamaño del conjunto de datos: $n = 10^2, \ldots, 10^5$.
- Número de variables: $n\_vars = 10$.
- Para **LIBSVM**:
  - Parámetro de regularización: $C = 1$.
  - Cantidad de caché: $m = 100$ megabytes.
  - *Reducción*: $shrinking = 1$.
  - Parámetro del Kernel RBF: $\gamma = 0.1$.
- Para **SVM-SGD**:
  - Parámetro de regularización: $\lambda = 1$.
  - Número de *períodos*: $n\_epochs = 10$.

- Ejemplo con 2 variables y $n = 1000$.



Problemas simulados: Problema no linealmente separable

- Ejemplo con 2 variables y $n = 1000$.

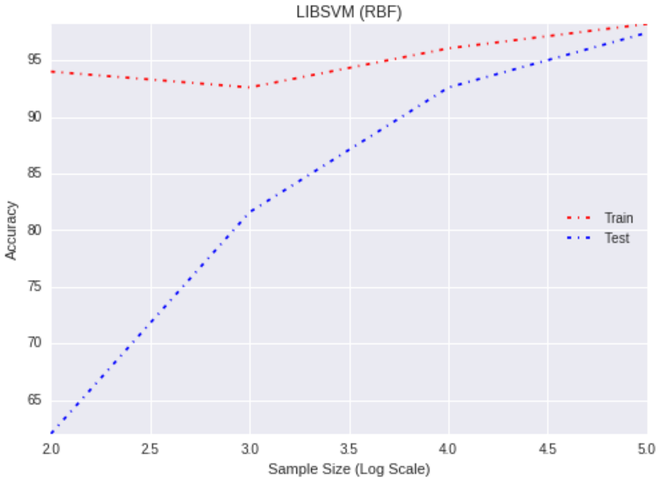


Problemas simulados: Curvas de aprendizaje

Problema lineal	10^2	10^3	10^4	10^5
SVM-SGD: Acc. Train	92.0	94.3	94.0	93.41
SVM-SGD: Acc. Test	93.34	93.95	93.74	93.26
SVM-SGD: Tiempo	0.0004	0.0019	0.0140	0.2325
LIBSVM (Lineal): Acc. Train	94.0	94.4	94.24	94.32
LIBSVM (Lineal): Acc. Test	90.99	93.98	94.18	94.19
LIBSVM (Lineal): Tiempo	0.0004	0.0183	0.2370	548.17
LIBSVM (RBF): Acc. Train	100.0	99.8	98.93	98.76
LIBSVM (RBF): Acc. Test	94.51	97.87	98.34	98.54
LIBSVM (RBF): Tiempo	0.0025	0.0051	0.1487	252.74

Problema no lineal	10^2	10^3	10^4	10^5
SVM-SGD: Acc. Train	50.0	50.3	50.12	50.27
SVM-SGD: Acc. Test	49.7	50.3	49.7	50.3
SVM-SGD: Tiempo	0.0004	0.0019	0.0079	0.1151
LIBSVM (RBF): Acc. Train	94.0	92.6	96.05	98.21
LIBSVM (RBF): Acc. Test	62.02	81.59	92.59	97.43
LIBSVM (RBF): Tiempo	0.0013	0.0456	0.3140	271.71

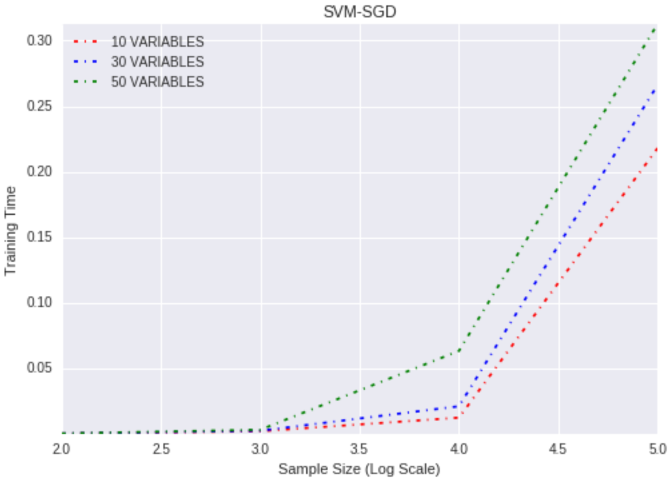
Problemas simulados: Curvas de aprendizaje



Problemas simulados: Número de variables

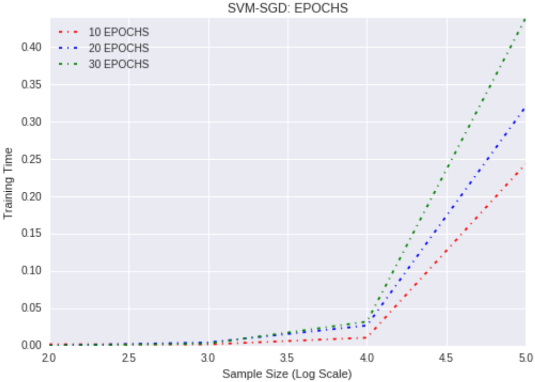
Problema lineal ($n = 10^5$)	10	30	50
SVM-SGD: Acc. Train	93.41	92.62	93.32
SVM-SGD: Acc. Test	93.26	92.43	93.81
SVM-SGD: Tiempo	0.2179	0.2661	0.3126
LIBSVM (Lineal): Acc. Train	94.32	94.32	94.32
LIBSVM (Lineal): Acc. Test	94.19	94.19	94.19
LIBSVM (Lineal): Tiempo	539.85	525.44	546.86
LIBSVM (RBF): Acc. Train	98.76	98.76	98.76
LIBSVM (RBF): Acc. Test	98.54	98.54	98.54
LIBSVM (RBF): Tiempo	258.40	248.30	253.72

Problemas simulados: Número de variables



Problemas simulados: Número de *períodos*

Problema lineal $(n = 10^5)$	10	20	30
SVM-SGD: Acc. Train	93.41	93.92	93.89
SVM-SGD: Acc. Test	93.26	93.95	93.94
SVM-SGD: Tiempo	0.2438	0.3204	0.4390



Problemas simulados: *Reducción*

Problema lineal ($n = 10^5$)	Con *reducción*	Sin *reducción*
LIBSVM (Lineal): Acc. Train	94.32	94.32
LIBSVM (Lineal): Acc. Test	94.19	94.19
LIBSVM (Lineal): Tiempo	555.16	1268.01

Problema no lineal ($n = 10^5$)	Con *reducción*	Sin *reducción*
LIBSVM (RBF): Acc. Train	98.21	98.22
LIBSVM (RBF): Acc. Test	97.30	97.15
LIBSVM (RBF): Tiempo	282.44	351.30

Problemas reales

Dataset	Referencia	Núm. observaciones	Núm. Variables
ijcnn1	[16]	49990	22
cod-rna	[17]	59535	8
a9a	[18]	32561	123
w8a	[18]	49749	300

	SVM-SGD	SVM-SGD	SVM-SGD	LIBSVM	LIBSVM	LIBSVM
Dataset	Acc. Train	Acc. Test	Tiempo	Acc. Train	Acc. Test	Tiempo
ijcnn1	90.24	90.46	0.06	94.11	94.19	29.32
cod-rna	83.47	83.42	0.05	89.79	89.09	145.31
a9a	76.05	75.35	0.03	86.72	83.89	57.78
w8a	97.05	96.90	0.05	99.06	98.71	68.65

Dataset	Ratio Acc. Train	Ratio Acc. Test	Ratio Tiempo
ijcnn1	.9588	.9603	.0020
cod-rna	.9296	.9363	.0003
a9a	.8769	.8982	.0005
w8a	.9797	.9816	.0007

Conclusiones I

- **SVM-SGD** es muy eficiente computacionalmente.
- Para problemas con estructura lineal, **SVM-SGD** es muy superior respecto a **LIBSVM** logrando resultados similares en predicción.
- **LIBSVM** es dependiente de la cantidad de memoria disponible, mientras que utiizar **SVM-SGD** puede utilizarse aún teniendo memoria limitada.
- Utilizar **LIBSVM** con un Kernel no lineal lleva a mejores resultados en predicción, con el costo de un mucho mayor tiempo de entrenamiento.
- En otro caso, si se privilegia la capacidad de predicción y se cuenta con la memoria suficiente para el volumen de datos, es preferible utilizar **LIBSVM** con un Kernel no lineal.

Conclusiones II

- **SVM-SGD** y **LIBSVM** no son paralelizables *per se* ya que los algoritmos de optimización que utilizan son secuenciales. La búsqueda de los parámetros, en ambos casos, puede ser realizada en paralelo.
- Pueden desarrollarse implementaciones del método MSV con el Kernel lineal y con Kernels no lineales, basadas en métodos de puntos interiores, que son altamente paralelizables.

Bibliografía I

- BISHOP, C. (2010), *Pattern Recognition and Machine Learning*. Springer.
- HASTIE, T. *et al.* (2008), *The elements of Statistical Learning*. Springer.
- VAPNIK, VLADIMIR AND CORTES, CORINNA (1995), *Support Vector Networks*. Machine Learning, 20.
- VAPNIK, VLADIMIR (1982), *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics.
- JOACHIMS, THORSTEN (1999), *Making Large Scale SVM Learning Practical*. Advances in Kernel Methods. MIT Press.

Bibliografía II

- Chen, Pai-Hsuen *et al.* (2006), *A Study on SMO-type Decomposition Methods For Support Vector Machines*. IEEE Transactions on Neural Networks.
- Steinwart, Ingo and Scovel, Clint (2004), *Fast rates for support vector machines using gaussian kernels*. Technical Report.
- Karatzoglou, A. *et al.* (2013), *kernlab: An S4 Package for Kernel Methods in R*. The Comprehensive R Archive Network.
- Shiloh, R. (2007), *Support Vector Machines for Classification and Regression*. M.Sc. Thesis, McGill University.

ITAM

Bibliografía III

- RIFKIN, R. (2002), *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. Ph. D. Thesis, MIT.
- BOTTOU, LEON AND LIN, CHIH-JEN (2007), *Support Vector Machines Solvers*. MIT Press.
- BOTTOU, LEON (2010), *Large Scale Machine Learning With Stochastic Gradient Descent*. Documentación y software disponible en http://leon.bottou.org/projects/sgd
- CHANG, CHIH-CHUNG AND LIN, CHIH-JEN (2007), *LIBSVM: A library for Suppport Vector Machines*. Documentación y software disponible en http://www.csie.ntu.edu.tw/~cjlin/libsvm

Bibliografía IV

- Shalev-Shwartz, Shai *et al.* (2013), *Pegasos: Primal Estimated Sub-Gradient Solver for SVM*. Proceedings of the 24th International Conference on Machine Learning.
- Krishna, Aditya, *Large Scale Support Vector Machines: Algorithms and Theory*.
- Prokhorov, Danil (2001), *Slide presentation in IJCNN 2001, Ford Research Laboratory* in IJCNN 2001 Neural Network Competition.
- Uzilov, A. *et al.* (2006), *Detection of Non-Coding RNAs on the Basis of Predicted Secondary Structure Formation Free Energy Change.*. BMC Bioinformatics.

Bibliografía V

- PLATT, JOHN (1998), *Fast Training of Support Vector Machines Using Sequential Minimal Optimization* in Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, Advances in Kernel Methods - Support Vector Learning. MIT Press.
- MERCER, J. (1909), *Functions of Positive and Negative Type and Their Connection With the Theory of Integral Equations*. Philosophical Transactions of the Royal Society.
- ARONSZAJN, N. (1950), *Theory of Reproducing Kernels*. Transactions of the American Mathematical Society.

Bibliografía VI

Gondzio, J. and Woodsend, K. (2009), *Exploiting Separability in Large-Scale Linear Support Vector Machine Training*. Technical Report. The University of Edinburgh.

Gondzio, J. and Woodsend, K. (2007), *Parallel Support Vector Machine Training with Nonlinear Kernels*. Technical Report. The University of Edinburgh.