UAM
Universidad Autónoma de Madrid

Advanced Kernel Methods for Multi-Task Learning

Tesis dirigida por José Dorronsoro y Carlos Alaíz

Carlos Ruiz Pastor

20 de abril de 2023

UAM
Universidad Autónoma de Madrid

UAM
Universidad Autónoma de Madrid

Índice

0.

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

Aprendizaje automático

- En el aprendizaje supervisado tenemos
  - un espacio de entrada $\mathcal{X}$
  - un espacio de salida $\mathcal{Y}$
  - y una distribución $P(x, y)$ (desconocida) sobre $\mathcal{X} \times \mathcal{Y}$
- Queremos estimar la relación entre x e y:

$$f : \mathcal{X} \to \mathcal{Y}$$

- Consideramos funciones h un espacio de hipótesis H
- Dada una función $h : \mathcal{X} \to \mathcal{Y}$, definimos una función de pérdida como

$$\begin{array}{rcl} \ell : \mathcal{Y} \times \mathcal{Y} & \to & [0, \infty) \\ (y, h(x)) & \to & \ell(y, h(x)) \end{array}$$

Riesgo esperado

- Definimos el Riesgo Esperado como

$$R_P(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h(x)) dP(x, y), \;\; h \in \mathcal{H}$$

- El objetivo es minimizar el Riesgo Esperado:

$$h^* = \arg\min_{h\in\mathcal{H}} \left\{ R_P(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h(x)) dP(x, y) \right\}$$

- Pero la distribución $P(x, y)$ es desconocida

Riesgo empírico

- En su lugar tenemos n muestras de $P(x, y)$:

$$D = \{(x_i, y_i) \sim P(x, y),\ i = 1, \ldots, n\}$$

- Definimos el Riesgo Empírico como

$$\hat{R}_D(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i))$$

- Una estrategia común es minimizar el Riesgo Empírico Regularizado:

$$\arg\min_{h \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i)) + \Omega(h) \right\}$$

Aprendizaje multitarea

- En aprendizaje multitarea con T tareas tenemos
  - un espacio de entrada $\mathcal{X}$,
  - un espacio de salida $\mathcal{Y}$,
  - y T distribuciónes $\boldsymbol{P} = (P_1, \ldots, P_T)$ (desconocidas) sobre $\mathcal{X} \times \mathcal{Y}$
- Tenemos que estimar T hipótesis $\boldsymbol{h} = (h_1, \ldots, h_T) \in \mathcal{H}^T$
- El Riesgo Esperado multitarea es

$$R_{\boldsymbol{P}}(\boldsymbol{h}) = \sum_{r=1}^{T} \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h_r(x)) dP_r(x, y)$$

Riesgo empírico multitarea

- Tenemos una muestra multitarea

$$\boldsymbol{D} = \bigcup_{r=1}^{T} \{(x_i^r, y_i^r) \sim P_r(x, y),\ i = 1, \ldots, m_r\}$$

- El Riesgo Empírico multitarea es

$$\hat{R}_{\boldsymbol{D}}(\boldsymbol{h}) = \sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(y_i^r, h_r(x_i^r))$$

- Minimizamos el Riesgo Empírico Regularizado multitarea

$$\arg\min_{\boldsymbol{h} \in \mathcal{H}^T} \left\{ \sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(y_i^r, h_r(x_i^r)) + \Omega(\boldsymbol{h}) \right\}$$

CTL vs ITL vs MTL

- Hay tres opciones para minimizar el Riesgo Regularizado multitarea
  - Aprendizaje común (CTL): se usa un modelo común para todas las tareas

$$\arg\min_{h\in\mathcal{H}} \left\{ \sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(y_i^r, h(x_i^r)) + \Omega(h) \right\}$$

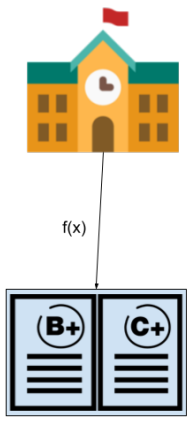
  - Aprendizaje independiente (ITL): se usan modelos independientes en cada tarea

$$\arg\min_{\boldsymbol{h}\in\mathcal{H}^T} \left\{ \sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(h_r(x_i^r), y_i^r) + \sum_{r=1}^{T} \Omega_r(h_r) \right\}$$

  - Aprendizaje multitarea (MTL): se usan modelos específicos que comparten información

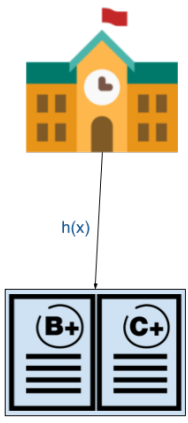
Aprendizaje estándar: una tarea

1. Introducción



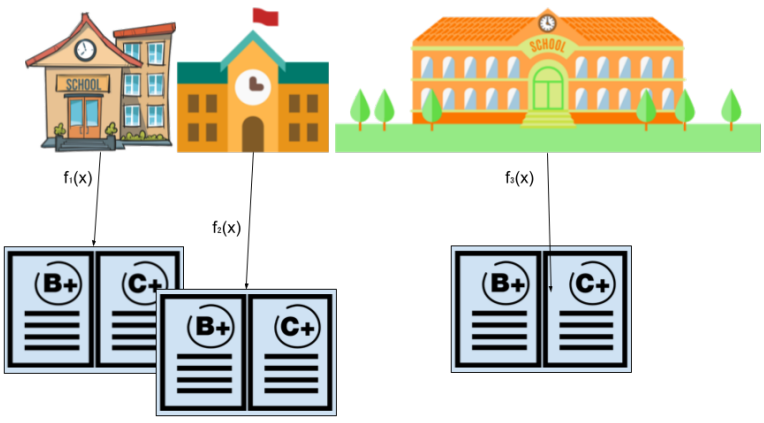
Aprendizaje estándar: una tarea

1. Introducción



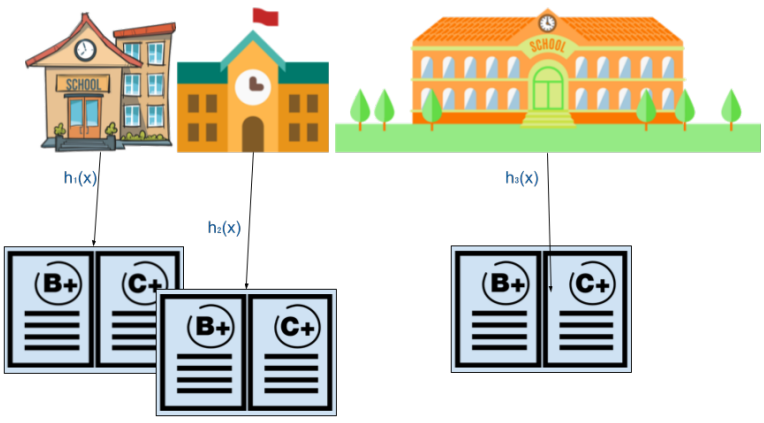
Aprendizaje estándar: varias tareas

1. Introducción



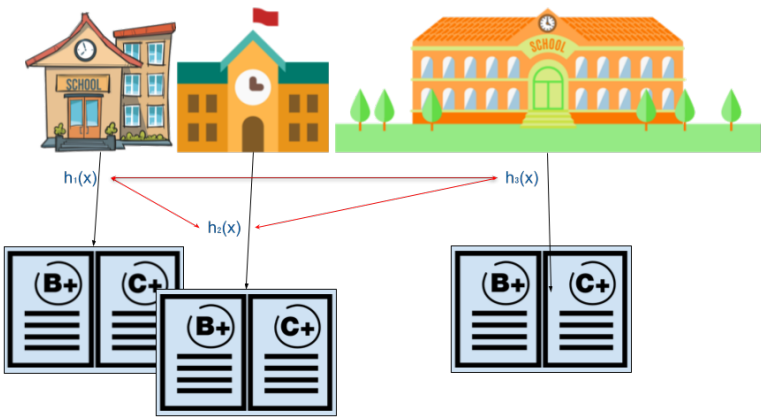
Aprendizaje estándar: varias tareas

1. Introducción



Aprendizaje multitarea

1. Introducción



Aprendizaje multitarea: estrategias

- Modelos basados en características (más comunes en redes neuronales)

$$\sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(g_r \circ f(x_i^r), y_i^r) + \Omega(f) + \Omega(g_1, \ldots, g_T)$$

- Modelos basados en regularización (más comunes con modelos lineales)

$$\sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(h_r(x_i^r), y_i^r) + \Omega(h_1, \ldots, h_T), \ \Omega(h_1, \ldots, h_T) \neq \sum_{r=1}^{T} \Omega_r(h_r)$$

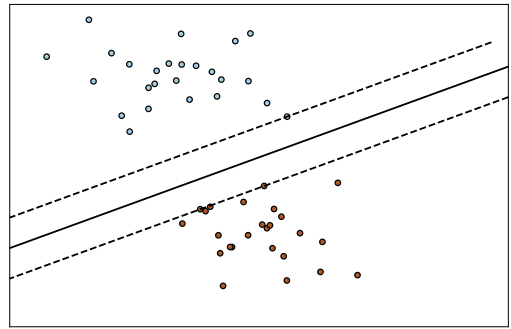
- Modelos basados en combinación (más comunes con SVMs)

$$\sum_{r=1}^{T} \frac{1}{m_r} \sum_{i=1}^{m_r} \ell(g(x_i^r) + g_r(x_i^r), y_i^r) + \Omega(g) + \Omega(g_1, \ldots, g_T)$$

Máquinas de vectores soporte (SVM)

1. Introducción

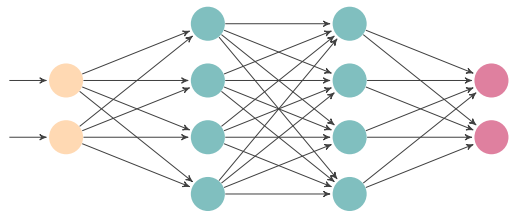
- Se definen usando problemas convexos
- Se puede aplicar el truco del kernel
- Tienen una limitación computacional
- *Mejores para problemas pequeños*



Redes Neuronales (NN)

1. Introducción

- Son muy flexibles
- Escalan linealmente con el tamaño de los problemas que se usan
- Necesitan muchos datos
- *Mejores para problemas grandes*



Objetivos y contribuciones

- **Objetivos**
  - Formulación más interpretable para el aprendizaje MT con SVM
  - Modelos neuronales MT alternativos basados en combinaciones
  - Método para aprender la relación entre tareas
- **Contribuciones**
  - Revisión del estado del arte sobre aprendizaje multitarea
  - Combinación convexa para SVM MT[1]
  - Combinación convexa para redes neuronales MT[2]
  - Regularización laplaciana adaptativa para SVM MT

[1] https://github.com/carlosruizp/mtlskl

[2] https://github.com/carlosruizp/convexMTLPyTorch

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

Problema Primal - SVM

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq p_i - \xi_i, \\ & \xi_i \geq 0 \end{aligned}$$

- Se puede ver que esta formulación es equivalente a
  - Máquina de Vectores Soporte para Clasificación (SVC): $p_i = 1$ para $i = 1, \ldots n$
  - Máquina de Vectores Soporte para Regresión (SVR): se duplican los patrones
    - $y_i = 1,\ p_i = t_i - \epsilon$, en la primera mitad
    - $y_i = -1,\ p_i = -t_i - \epsilon$, en la segunda mitad

Problema Dual - SVM

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top Q \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{p} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C \end{aligned}$$

- Aquí Q es la matriz de kernel (con etiqueta)
  - SVM lineal: $Q_{ij} = y_i y_j \langle x_i, x_j \rangle$
  - SVM no lineal: $Q_{ij} = y_i y_j k(x_i, x_j) = y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$

Formulación aditiva con SVM

- Una manera de implementar el MTL es combinar una parte común y otras específicas
- Fue propuesta[3] incialmente para SVM lineales:

$$h_r(\cdot) = \langle w + v_r, \cdot \rangle + b_r$$

- Fue extendida al caso no lineal[4]:

$$h_r(\cdot) = \langle w, \phi(\cdot) \rangle + \langle v_r, \phi_r(\cdot) \rangle + b_r$$

[3] Theodoros Evgeniou y Massimiliano Pontil. "Regularized multi–task learning". En: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, págs. 109-117.

[4] Feng Cai y Vladimir Cherkassky. "SVM+ regression and multi-task learning". En: *International Joint Conference on Neural Networks*. 2009, págs. 418-424.

Formulación aditiva para SVM MT

Problema primal - SVM MT aditiva

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{v},\boldsymbol{b},\xi}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{\mu}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(\langle w, \phi(x_i^r)\rangle + \langle v_r, \phi_r(x_i^r)\rangle + b_r) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0;\ i = 1,\dots,m_r,\ r = 1,\dots,T \end{aligned}$$

- El parámetro μ (junto con C) regula la influencia de cada parte:
  - $\mu \to \infty$: modelos independientes (ITL)
  - $C \to 0,\ \mu \to 0$: modelo común (CTL)
- Tenemos la transformación común ϕ y las específicas ϕ_r

Formulacion convexa con SVM

- Proponemos[5] la siguiente formulación convexa para el aprendizaje multitarea:

$$h_r(\cdot) = \lambda_r \left\{ \langle w, \phi(\cdot) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(\cdot) \rangle + d_r \right\}, \ \lambda_r \in [0, 1]$$

- Los hiperparámetros λ_r, en lugar de μ, regulan la influencia de cada parte
  - $\lambda_1, \ldots, \lambda_T = 0$: ITL
  - $\lambda_1, \ldots, \lambda_T = 1$: CTL
- Extendemos esta formulación[6] y desarrollamos tres variantes de SVM

[5] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "A Convex Formulation of SVM-Based Multi-task Learning". En: *HAIS Proceedings*. Vol. 11734. Springer, 2019, págs. 404-415.

[6] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex formulation for multi-task L1-, L2-, and LS-SVMs". En: *Neurocomputing* 456 (2021), págs. 599-608.

Formulación convexa para SVM MT

Problema primal - SVM MT convexa

$$\min_{w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r\left(\lambda_r\left\{\langle w,\phi(x_i^r)\rangle + b\right\} + (1-\lambda_r)\left\{\langle v_r,\phi_r(x_i^r)\rangle + d_r\right\}\right) \geq p_i^r - \xi_i^r,$$

$$\xi_i^r \geq 0,\ i = 1,\ldots,m_r,\ r = 1,\ldots,T$$

- Se sustituye μ por los hiperparámetros λ_r
- El hiperparámetro C no interviene en el grado de interdependencia de los modelos

Formulación convexa para SVM MT

Problema dual - SVM MT convexa

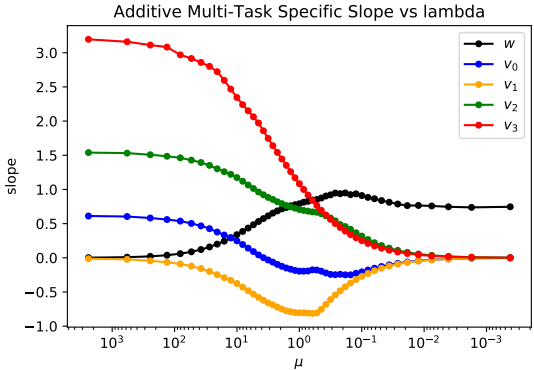
$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\intercal \left(\Lambda Q \Lambda + (I_n - \Lambda) K (I_n - \Lambda)\right) \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C;\ i = 1, \ldots, m_r,\ r = 1, \ldots, T, \\ & \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0;\ r = 1, \ldots, T \end{aligned}$$

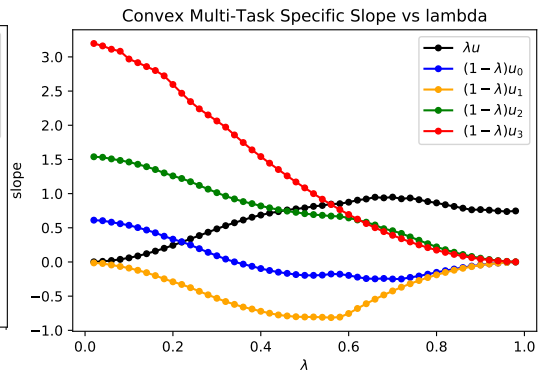
- Usamos la matriz $\Lambda = \mathrm{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T})$
- La matriz Q es común entre todas las tareas usando el kernel k correspondiente a ϕ
- La matriz K es diagonal por bloques, con los kernel k_r correspondientes a ϕ_r
- La función de kernel es:

$$\widehat{k}(x_i^r, x_j^s) = \lambda_r \lambda_s k(x_i^r, x_j^s) + \delta_{rs}(1 - \lambda_r)(1 - \lambda_s) k_r(x_i^r, x_j^s)$$

Equivalencia entre formulaciones aditiva y convexa

- En la tesis mostramos la equivalencia de forma teórica y la relación entre λ y μ





Alternativa: combinación de modelos preentrenados

- Como comprobación, consideramos[7] la combinación convexa de modelos preentrenados
  - modelo común $g(\cdot)$ ya entrenado
  - modelos específicos $g_r(\cdot)$ ya entrenados
- Minimizamos el riesgo eligiendo los hiperparámetros $\lambda_1, \ldots, \lambda_T$ óptimos

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r), y_i^r),$$

- Consideramos las pérdidas:
  - Cuadrática y Absoluta (regresión)
  - Hinge y Hinge Cuadrática (clasificación)

[7] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex formulation for multi-task L1-, L2-, and LS-SVMs". En: *Neurocomputing* 456 (2021), págs. 599-608.

Experimentos: modelos

- **Common Task Learning SVM (CTL)**: Un único modelo SVM que es común para todas las tareas
- **Independent Task Learning SVM (ITL)**: Un modelo SVM independiente para cada tarea
- **Direct Convex Combination of SVMs (CMB)**: Una combinación convexa de los mejores CTL e ITL preentrenados
- **Convex Multi-Task Learning SVM (MTL)**: Un modelo multitarea con la formulación convexa basado en la SVM

UAM
Universidad Autónoma de Madrid

Experimentos: problemas

2. Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte

Problema	Tamaño	Dimensión	N° tareas	Tam. tarea medio	Tam. tarea mín.	Tam. tarea máx.
majorca	15 330	765	14	1095	1095	1095
tenerife	15 330	765	14	1095	1095	1095
california	19 269	9	5	3853	5	8468
boston	506	12	2	253	35	471
abalone	4177	8	3	1392	1307	1527
crime	1195	127	9	132	60	278
binding	32 302	184	47	687	59	3089
landmine	14 820	10	28	511	445	690
adult_(G)	48 842	106	2	24 421	16 192	32 650
adult_(R)	48 842	103	5	9768	406	41 762
adult_(G, R)	48 842	101	10	4884	155	28 735
compas_(G)	3987	11	2	1993	840	3147
compas_(R)	3987	9	4	997	255	1918
compas_(G, R)	3987	7	8	498	50	1525

Experimentos: hiperparametrización

- Hacemos una búsqueda en rejilla con una validación cruzada
- Usamos tres particiones estratificadas por tareas
- Por limitaciones computacionales nos restringimos a la búsqueda de tres hiperparámetros para la búsqueda en rejilla

	Rejilla	CTL-L1,2	ITL-L1,2	MTL-L1,2	CTL-LS	ITL-LS	MTL-LS
C	$\{4^k : -2 \leq k \leq 6\}$	CV	CV	CV	CV	CV	CV
ϵ	$\{\frac{\sigma}{4^k} : 1 \leq k \leq 6\}$	CV	CV	CV	-	-	-
γ_c	$\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$	CV	-	CTL-L1,2	CV	-	CTL-LS
γ_s^r	$\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$	-	CV	ITL-L1,2	-	CV	ITL-LS
λ	$\{0{,}1k : 0 \leq k \leq 10\}$	-	-	CV	-	-	CV

Experimentos: resultados de regresión

2. Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte

	maj.	ten.	boston	california	abalone	crime
ITL-L1	5.087	5.743	2.341	36883.582	1.481	0.078
CTL-L1	5.175	5.891	**2.192**	41754.337	1.482	0.078
CMB-L1	**5.047**	**5.340**	2.239	36880.238	1.470	0.077
MTL-L1	5.050	5.535	2.206	**36711.383**	**1.454**	**0.074**
ITL-L2	4.952	**5.629**	2.356	37374.618	1.498	0.079
CTL-L2	5.193	6.107	**2.083**	42335.612	1.503	0.080
CMB-L2	4.869	5.963	2.089	37374.618	1.494	0.077
MTL-L2	**4.854**	5.784	2.089	**37202.603**	**1.482**	**0.077**
ITL-LS	4.937	5.649	2.204	37348.347	1.496	0.079
CTL-LS	5.193	6.005	**2.072**	42259.492	1.502	0.079
CMB-LS	4.977	**5.593**	2.081	37339.179	1.486	0.079
MTL-LS	**4.824**	5.754	2.077	**37231.043**	**1.478**	**0.076**

UAM
Universidad Autónoma de Madrid

Experimentos: resultados de clasificación

2. Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte

	comp_(G)	comp_(R)	comp_(G,R)	ad_(G)	ad_(R)	ad_(G,R)	landmine	binding
ITL-L1	0.625	**0.639**	0.630	**0.659**	0.653	0.657	0.231	0.867
CTL-L1	0.623	0.638	0.638	0.657	0.650	0.653	0.255	0.901
CMB-L1	0.616	0.638	0.638	0.658	0.650	0.653	**0.270**	0.901
MTL-L1	**0.627**	0.636	**0.640**	**0.659**	**0.655**	**0.659**	0.242	**0.907**
ITL-L2	0.636	0.623	0.607	**0.668**	**0.666**	**0.668**	0.256	0.867
CTL-L2	**0.640**	0.647	**0.651**	0.665	0.661	0.659	**0.270**	0.903
CMB-L2	0.629	0.640	0.645	0.666	0.662	0.661	**0.270**	0.903
MTL-L2	0.634	**0.651**	0.650	**0.668**	**0.666**	**0.668**	0.263	**0.909**
ITL-LS	**0.631**	0.622	0.608	**0.659**	**0.659**	**0.660**	0.243	0.867
CTL-LS	0.628	**0.644**	**0.649**	0.650	0.653	0.647	0.230	0.853
CMB-LS	0.630	0.635	0.642	0.657	0.658	0.654	0.238	0.873
MTL-LS	0.630	0.641	0.648	**0.659**	**0.659**	0.659	**0.257**	**0.906**

Experimentos: predicción de energía renovable

- Aplicamos la SVM MT convexa a la predicción de energía renovable
- Para energía solar consideramos las definiciones de tareas:
  - season
  - hour
- Para energía eólica consideramos las definiciones de tareas:
  - velocity
  - angle
  - timeOfDay
- También probamos la combinación de tareas, e.g. (season, hour)

UAM
Universidad Autónoma de Madrid

Experimentos: predicción de energía solar

2. Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte

	MAE				MSE				λ^*
	MWh	%	rank	Wil.	MWh^2	‰	rank	Wil.	
majorca									
CTL-L1	5.265	7.265	(6)	(4)	59.322	112.985	(6)	(4)	-
(season)_ITL-L1	5.305	7.384	(7)	(5)	59.591	113.498	(7)	(4)	-
(season)_MTL-L1	**4.884**	**6.740**	**(1)**	**(1)**	53.222	101.366	(2)	(3)	0.4
(hour)_ITL-L1	5.083	7.015	(4)	(2)	54.540	103.877	(3)	(3)	-
(hour)_MTL-L1	4.957	6.840	(2)	(1)	**52.614**	**100.208**	**(1)**	**(1)**	0.3
(hour, season)_ITL-L1	5.250	7.251	(5)	(3)	57.927	110.328	(5)	(4)	-
(hour, season)_MTL-L1	5.038	6.952	(3)	(2)	54.601	103.992	(4)	(3)	0.3
tenerife									
CTL-L1	5.786	5.373	(5)	(5)	88.323	76.174	(5)	(5)	-
(season)_ITL-L1	5.930	5.545	(6)	(5)	97.454	84.611	(6)	(6)	-
(season)_MTL-L1	5.579	5.181	(4)	(2)	86.227	74.366	(3)	(3)	0.8
(hour)_ITL-L1	5.403	5.018	(2)	(2)	86.686	74.762	(4)	(4)	-
(hour)_MTL-L1	**5.376**	**4.993**	**(1)**	(1)	**84.207**	**72.624**	**(1)**	(1)	0.7
(hour, season)_ITL-L1	6.025	5.554	(7)	(6)	104.536	90.297	(7)	(7)	-
(hour, season)_MTL-L1	5.494	5.102	(3)	(3)	85.440	73.687	(2)	(2)	0.7

UAM
Universidad Autónoma de Madrid

Experimentos: predicción de energía eólica

2. Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte

	MAE			MSE			λ^*
	MWh	rank	Wil.	MWh^2	rank	Wil.	
CTL-L1	**6.132**	**(1)**	**(1)**	90.228	(2)	(2)	-
(velocity)_ITL-L1	6.211	(7)	(3)	93.363	(7)	(3)	-
(velocity)_MTL-L1	6.208	(6)	(3)	93.199	(6)	(3)	0
(timeOfDay)_ITL-L1	6.283	(9)	(4)	93.594	(9)	(4)	-
(timeOfDay)_MTL-L1	**6.132**	**(1)**	**(1)**	90.228	(2)	(2)	1
(timeOfDay, velocity)_ITL-L1	6.341	(11)	(4)	97.250	(11)	(5)	-
(timeOfDay, velocity)_MTL-L1	6.312	(10)	(4)	94.774	(10)	(4)	0.4
(timeOfDay, angle)_ITL-L1	6.266	(8)	(4)	93.517	(8)	(4)	-
(timeOfDay, angle)_MTL-L1	**6.132**	**(1)**	**(1)**	90.228	(2)	(2)	1
(timeOfDay, angle, velocity)_ITL-L1	6.410	(12)	(4)	102.031	(12)	(6)	-
(timeOfDay, angle, velocity)_MTL-L1	**6.132**	**(1)**	**(1)**	90.228	(2)	(2)	1
(angle)_ITL-L1	6.170	(4)	(3)	91.586	(4)	(3)	-
(angle)_MTL-L1	6.135	(2)	(2)	**90.026**	**(1)**	**(1)**	0.9
(angle, velocity)_ITL-L1	6.173	(5)	(3)	92.529	(5)	(3)	-
(angle, velocity)_MTL-L1	6.168	(3)	(3)	90.990	(3)	(3)	0.7

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

Redes neuronales MT

- Los SVM ofrecen propiedades deseables como
  - convexidad
  - dualidad
  - truco del kernel

  pero tienen una limitación computacional
- Las redes neuronales son mejores alternativas para problemas grandes
- Existen arquitecturas neuronales específicas para algunos tipos de datos
  - imágenes
  - texto
  - sonido

Redes neuronales MT: *hard sharing*

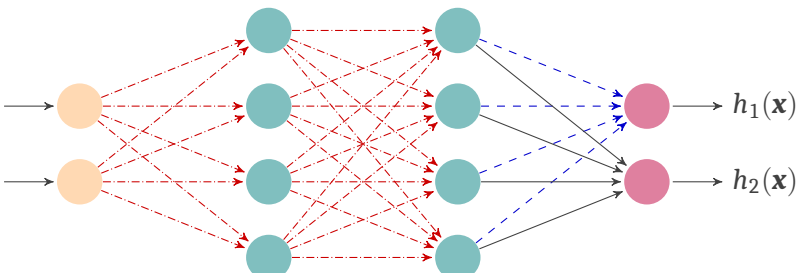
- La manera más común de adaptar las redes neuronales es el *hard sharing*[8]
  - capas compartidas
  - capas específicas
- El modelo simplificado se puede expresar como:

$$h_r(\cdot) = \langle w_r, f(\cdot; \Theta) \rangle + d_r$$

  - w_r, d_r son los parámetros de las capas de salida específicas
  - Θ son los parámetros de las capas ocultas compartidas
- Se comparte la misma representación en todas las tareas

[8] Rich Caruana. "Multitask Learning". En: *Mach. Learn.* 28.1 (1997), págs. 41-75.

Ejemplo de *hard sharing* para dos tareas



Formulación Convexa para redes neuronales MT

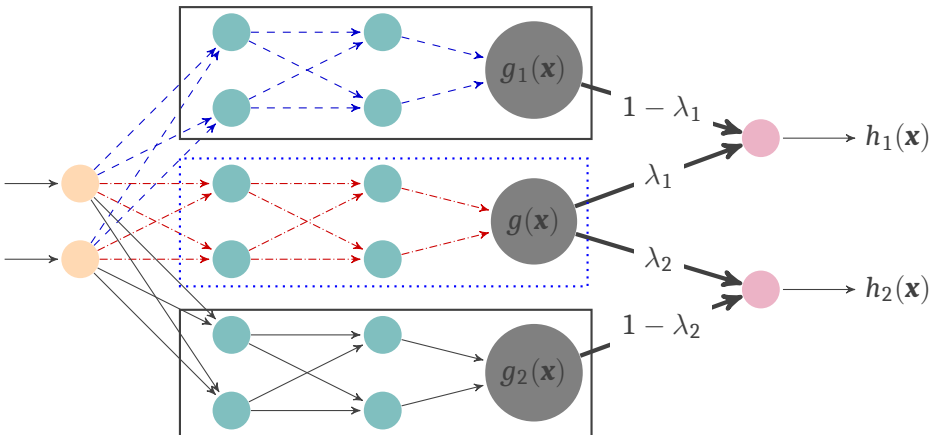
- Proponemos[9] la formulación convexa para redes neuronales MT
- Los modelos son:

$$h_r(\cdot) = \lambda_r\{\langle w, f(\cdot;\Theta)\rangle + b\} + (1-\lambda_r)\{\langle w_r, f_r(\cdot;\Theta_r)\rangle + d_r\}$$

  - w, Θ son los parámetros de la red común
  - w_r, Θ_r son los parámetros de las redes específicas
- No se comparte la representación; se combinan una parte común y partes específicas

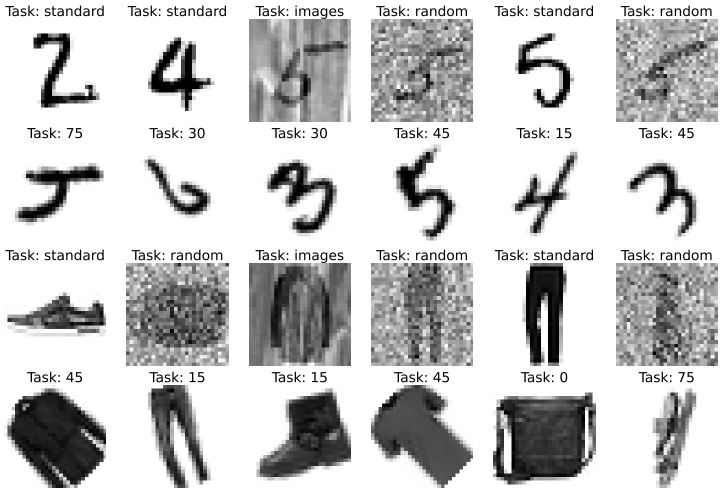
[9] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Multi-Task Learning with Neural Networks". En: *HAIS Proceedings*. Vol. 13469. Springer, 2022, págs. 223-235.

Ejemplo de formulación convexa para dos tareas



Experimentos: conjuntos de datos

3. Formulación convexa para aprendizaje multitarea: redes neuronales



Experimentos: resultados

	var-MNIST	rot-MNIST	var-FMNIST	rot-FMNIST
		accuracy		
ctlNN	0.964	0.973	0.784	0.834
itlNN	0.968	0.981	0.795	0.873
hsmtlNN	0.971	0.980	0.770	0.852
cvxmtlNN	**0.974** ($\lambda^* = 0.6$)	**0.984** ($\lambda^* = 0.8$)	**0.812** ($\lambda^* = 0.6$)	**0.880** ($\lambda^* = 0.6$)
		categorical cross-entropy		
ctlNN	1.274 ± 0.143	1.145 ± 0.039	2.369 ± 0.183	1.757 ± 0.075
itlNN	1.072 ± 0.029	0.873 ± 0.058	2.356 ± 0.130	1.598 ± 0.042
hsmtlNN	1.087 ± 0.253	0.898 ± 0.073	3.067 ± 0.888	1.888 ± 0.075
cvxmtlNN	$\mathbf{0.924 \pm 0.024}$ ($\lambda^* = 0.6$)	$\mathbf{0.831 \pm 0.029}$ ($\lambda^* = 0.8$)	$\mathbf{2.147 \pm 0.090}$ ($\lambda^* = 0.6$)	$\mathbf{1.482 \pm 0.063}$ ($\lambda^* = 0.6$)

- Todos los modelos se entrenan con el algoritmo *AdamW*. CV para hiperparámetros:
  - α (weight decay)
  - λ (en el modelo convexo)

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- **Laplaciano adaptativo para aprendizaje multitarea**
- Conclusiones y trabajo futuro

Aprendizaje multitarea con regularización laplaciana

4. Laplaciano adaptativo para aprendizaje multitarea

- La formulación convexa asume que hay una parte común a todas las tareas
- Por otra parte, la definición de tareas puede ser arbitraria
- Una alternativa para el aprendizaje MT es usar una regularización laplaciana
  - comprobar la definición de las tareas
  - medir relaciones entre tareas
  - mejorar modelos

Aprendizaje multitarea con regularización laplaciana

4. Laplaciano adaptativo para aprendizaje multitarea

- Consideramos un grafo donde
  - Los nodos representan tareas
  - Las aristas y sus pesos representan las relaciones entre las tareas
- La matriz de adyacencia A tiene los pesos de las aristas
- La matriz de grados D es una matriz diagonal donde

$$D_{rr} = \sum_{s=1}^{T} A_{rs}$$

- La matriz Laplaciana se define como $L = D - A$

Aprendizaje multitarea con regularización laplaciana

4. Laplaciano adaptativo para aprendizaje multitarea

- Dados los modelos para cada tarea definidos como

$$h_r(\cdot) = \langle w_r, \cdot \rangle + b_r,$$

añadimos la regularización

$$\sum_{r=1}^{T} \sum_{s=1}^{T} A_{rs} \|w_r - w_s\|^2$$

- Esta regularización se puede expresar como

$$\sum_{r=1}^{T} \sum_{s=1}^{T} A_{rs} \|w_r - w_s\|^2 = \sum_{r=1}^{T} \sum_{s=1}^{T} L_{rs} \langle w_r, w_s \rangle$$

- Proponemos combinar la formulación convexa con la regularización Laplaciana[10]

[10] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Graph Laplacian Multi-Task Learning SVM". En: *ICANN*. Vol. 12397. Springer, 2020, págs. 142-154.

Formulación convexa para SVM MT con laplaciano

Problema primal - SVM convexa con laplaciano

$$\min_{\boldsymbol{v},\boldsymbol{b},\boldsymbol{\xi},w} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}L_{rs}\langle v_r, v_s\rangle + \frac{1}{2}\sum_r \|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r(\lambda_r(\langle w, \phi(x_i^r)\rangle) + (1-\lambda_r)(\langle v_r, \psi(x_i^r)\rangle) + b_r) \geq p_i^r - \xi_i^r,$$

$$\xi_i^r \geq 0,\ i = 1,\dots,m_r,\ r = 1,\dots,T$$

- Los hiperparámetros λ_r regulan la influencia de cada parte:
  - $\lambda_1,\dots,\lambda_T = 0$: modelos independientes (ITL)
  - $\lambda_1,\dots,\lambda_T = 1$: modelo común (CTL)
- La matriz laplaciana L establece relaciones entre las partes específicas v_r

Formulación convexa para SVM MT con laplaciano

Problema dual - SVM convexa con laplaciano

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t \left(\Lambda Q \Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda)\right)\boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C,\ i = 1, \ldots, m_r, r = 1, \ldots, T, \\ & \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0,\ r = 1, \ldots, T \end{aligned}$$

- Usamos la matriz $\Lambda = \mathrm{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T})$
- La matriz Q es común entre todas las tareas usando el kernel k_ϕ correspondiente a ϕ
- La matriz $\tilde{Q}$ se define usando el kernel: $\widetilde{k}_\psi(x_i^r, x_j^s) = \left((\nu L + I_T)^{-1}\right)_{rs} k_\psi(x_i^r, x_j^s)$
- La función de kernel es: $\widehat{k}(x_i^r, x_j^s) = \lambda_r \lambda_s k_\phi(x_i^r, x_j^s) + (1 - \lambda_r)(1 - \lambda_s)\widetilde{k}_\psi(x_i^r, x_j^s)$

Algoritmo adaptativo para matriz de adyacencia

- La selección de la matriz de adyacencia A (y la respectiva L) es determinante
- Proponemos[11] un método para la selección automática de A, iterando los pasos
  - Minizamos en w, v_r con A fija
  - Minimizamos en A con los parámetros w, v_r fijos

[11] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Adaptive Graph Laplacian for Convex Multi-Task Learning SVM". En: *HAIS Proceedings*. Vol. 12886. Springer, 2021, págs. 219-230, Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Adaptive Graph Laplacian MTL L1, L2 and LS-SVMs". En: *Logic Journal of the IGPL* (in press).

Experimentos: problemas sintéticos

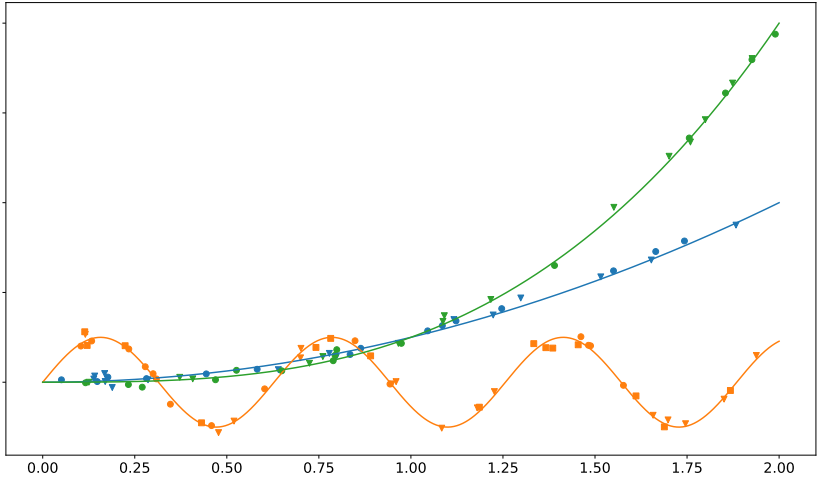
- Definimos problemas donde las tareas están agrupadas en clusters
  - Definimos τ tareas subyacentes (clusters) usando las funciones $f_r, r = 1, \ldots, \tau$
  - En cada una definimos T_r tareas virtuales, $r = 1, \ldots, \tau$ (las que ven los modelos)

	τ	f_r	T_r	Total tareas virtuales
regClusters0, clasClusters0	3	$f_1(x) = x^2$	2	7
		$f_2(x) = \sin(10x)$	3	
		$f_3(x) = x^3$	2	
regClusters1, clasClusters1	2	$f_1(x) = x^2$	5	7
		$f_2(x) = \sin(10x)$	2	
regClusters2, clasClusters2	2	$f_1(x) = \sin(10x)$	4	5
		$f_2(x) = x^3$	1	

UAM
Universidad Autónoma de Madrid

Experimentos: regClusterso

4. Laplaciano adaptativo para aprendizaje multitarea



Experimentos sintéticos: resultados regresión

4. Laplaciano adaptativo para aprendizaje multitarea

	regClusters0	regClusters1	regClusters2
		MAE	
CTL-L1	0.989	0.512	0.541
ITL-L1	0.221	0.212	0.159
MTL-L1	0.213	0.176	0.135
GLMTL-L1	0.212	0.173	0.138
AdapGLMTL-L1	**0.152**	**0.116**	**0.107**
CTL-L2	0.990	0.642	0.768
ITL-L2	0.213	0.201	0.154
MTL-L2	0.209	0.168	0.131
GLMTL-L2	0.204	0.169	0.131
AdapGLMTL-L2	**0.141**	**0.115**	**0.103**
CTL-LS	0.989	0.642	0.766
ITL-LS	0.212	0.209	0.149
MTL-LS	0.206	0.167	0.131
GLMTL-LS	0.207	0.169	0.132
AdapGLMTL-LS	**0.136**	**0.115**	**0.106**

Experimentos: matrices de adyacencia en regresión

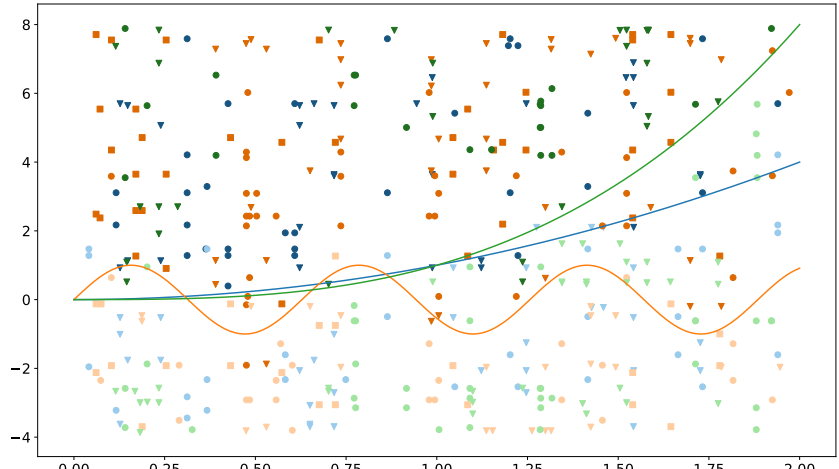
4. Laplaciano adaptativo para aprendizaje multitarea

- Matrices de regClusterso para L1, L2 y LS-SVM MT de laplaciano adaptativo

UAM
Universidad Autónoma de Madrid

Experimentos: problemas clasificación (clasClusterso)

4. Laplaciano adaptativo para aprendizaje multitarea



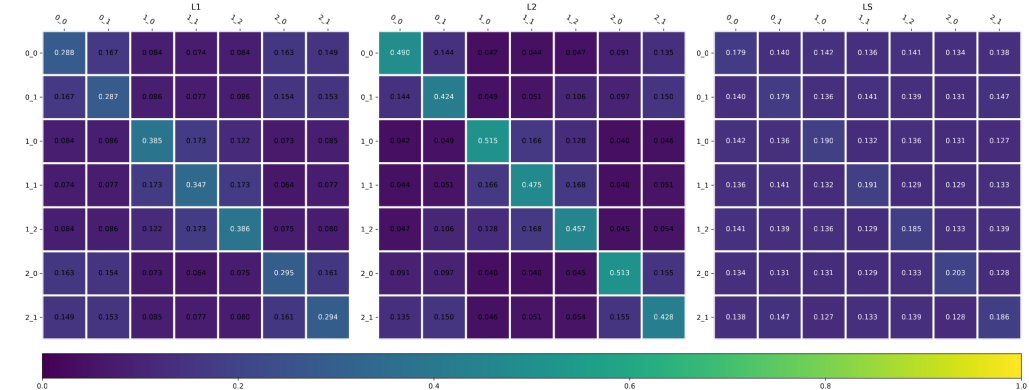
Experimentos: resultados clasificación

4. Laplaciano adaptativo para aprendizaje multitarea

	clasClusters0	clasClusters1	clasClusters2
		F1	
CTL-L1	0.901	0.912	0.904
ITL-L1	0.922	0.923	0.910
MTL-L1	**0.924**	0.925	0.914
GLMTL-L1	0.920	0.926	0.912
AdapGLMTL-L1	**0.924**	**0.929**	**0.916**
CTL-L2	0.904	0.912	0.906
ITL-L2	**0.928**	0.928	0.910
MTL-L2	0.925	0.927	0.913
GLMTL-L2	0.921	0.923	**0.915**
AdapGLMTL-L2	0.924	**0.929**	**0.915**
CTL-LS	0.895	0.908	0.894
ITL-LS	0.914	0.915	0.904
MTL-LS	0.917	0.917	**0.905**
GLMTL-LS	0.919	**0.921**	0.897
AdapGLMTL-LS	**0.920**	**0.921**	0.901

Experimentos: matrices de adyacencia en clasificación

4. Laplaciano adaptativo para aprendizaje multitarea



- Matrices de clasClusterso para L1, L2 y LS-SVM MT de laplaciano adaptativo

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

Conclusiones

- El Aprendizaje multitarea ofrece una serie de ventajas
- Es necesario desarrollar técnicas para crear un acoplamiento de los modelos
- Proponemos una formulación convexa para aprendizaje multitarea
  - SVM
  - Redes Neuronales
- Añadimos una regularización laplaciana para SVM
- Proponemos un método adaptativo para SVM con regularización laplaciana para aprender la relación entre tareas

Conclusiones: formulación convexa

- Formulación convexa con SVM:
  - Definimos SVM multitarea convexa
  - Analizamos la alternativa de combinación convexa de modelos preentrenados
  - Mostramos buenos resultados de las SVM MT convexas con varios problemas
  - Aplicamos estos modelos a predicción de energía solar y eólica
- Formulación convexa con redes neuronales:
  - Aplicamos esta formulación convexa también a redes neuronales
  - Mostramos mejores resultados que *hard sharing* en cuatro conjuntos de imágenes

Conclusiones: matriz de adyacencia adaptativa

- La formulación convexa asume que todas las tareas comparten una parte común
- Las tareas a veces están repetidas o no todas comparten la misma relación entre ellas
- Con la regularización laplaciana se pueden modelar distintas relaciones entre tareas definiendo una matriz de adyacencia adecuada
- Regularización laplaciana de grafo:
  - Definimos la SVM multitarea convexa con regularización laplaciana
  - Proponemos un algoritmo adaptativo para aprender la matriz de adyacencia
  - Obtenemos buenos resultados en problemas sintéticos y reales

Trabajo futuro

- Investigar métodos para la selección de hiperparámetros en los modelos MT
- Tratar los λ_r como parámetros de los modelos neuronales
- Aplicar la regularización laplaciana a modelos neuronales

UAM
Universidad Autónoma de Madrid

Advanced Kernel Methods for Multi-Task Learning

Gracias por su atención.

UAM
Universidad Autónoma de Madrid

Índice

5. Conclusiones y trabajo futuro

- Introducción
- Formulación convexa para aprendizaje multitarea: máquinas de vectores soporte
- Formulación convexa para aprendizaje multitarea: redes neuronales
- Laplaciano adaptativo para aprendizaje multitarea
- Conclusiones y trabajo futuro

Referencias

[1] Feng Cai y Vladimir Cherkassky. "SVM+ regression and multi-task learning". En: *International Joint Conference on Neural Networks*. 2009, págs. 418-424.

[2] Rich Caruana. "Multitask Learning". En: *Mach. Learn.* 28.1 (1997), págs. 41-75.

[3] Theodoros Evgeniou, Charles A. Micchelli y Massimiliano Pontil. "Learning Multiple Tasks with Kernel Methods". En: *J. Mach. Learn. Res.* 6 (2005), págs. 615-637.

[4] Theodoros Evgeniou y Massimiliano Pontil. "Regularized multi–task learning". En: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, págs. 109-117.

[5] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "A Convex Formulation of SVM-Based Multi-task Learning". En: *HAIS Proceedings*. Vol. 11734. Springer, 2019, págs. 404-415.

Referencias

[6] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Adaptive Graph Laplacian for Convex Multi-Task Learning SVM". En: *HAIS Proceedings*. Vol. 12886. Springer, 2021, págs. 219-230.

[7] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Adaptive Graph Laplacian MTL L1, L2 and LS-SVMs". En: *Logic Journal of the IGPL* (in press).

[8] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex formulation for multi-task L1-, L2-, and LS-SVMs". En: *Neurocomputing* 456 (2021), págs. 599-608.

[9] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Graph Laplacian Multi-Task Learning SVM". En: *ICANN*. Vol. 12397. Springer, 2020, págs. 142-154.

[10] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Multi-Task Learning with Neural Networks". En: *HAIS Proceedings*. Vol. 13469. Springer, 2022, págs. 223-235.

- ► Apéndice A: Combinación Convexa de Modelos Preentrenados
- ► Apéndice B: Aprendizaje MT Convexo con Redes Neuronales
- ► Apéndice C: Laplaciano de Grafo con Métodos de Kernel
- ► Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo
- ► Apéndice E: Extensiones a L2 y LS-SVM

Formulación Unificada Clasificación

7. Apéndice A: Combinación Convexa de Modelos Preentrenados

- Hinge loss (classification):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right]_+ .$$

- Squared hinge loss (classification):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right]_+^2 .$$

- Ambas se pueden expresar como:

$$\sum_{r=1}^{T} \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r), \text{ donde } c_i^r = y_i^r (g_r(x_i^r) - g(x_i^r)), \; d_i^r = 1 - y_i^r g_r(x_i^r)$$

Formulación Unificada Regresión

7. Apéndice A: Combinación Convexa de Modelos Preentrenados

- Absolute loss (regression):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left| y_i^r - \{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\} \right|.$$

- Squared loss (regression):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left(y_i^r - \{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\} \right)^2.$$

- Ambas se pueden expresar como:

$$\sum_{r=1}^{T} \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r), \text{ donde } c_i^r = g(x_i^r) - g_r(x_i^r), \; d_i^r = g_r(x_i^r) - y_i^r$$

Formulación Unificada

7. Apéndice A: Combinación Convexa de Modelos Preentrenados

- En todos los casos el riesgo lo podemos expresar como

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r)$$

- Como $\hat{R}_D(\lambda_1, \ldots, \lambda_T)$ es separable, tenemos en cada tarea el problema

$$\arg\min_{\lambda_r \in [0,1]} \mathcal{J}(\lambda_r) = \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r)$$

- Usando el Teorema de Fermat

$$\lambda^* = \arg\min_{0 \leq \lambda \leq 1} \mathcal{J}(\lambda) \iff (0 \in \partial\mathcal{J}(\lambda^*) \text{ y } \lambda^* \in (0,1)) \text{ o } \lambda^* = 0 \text{ o } \lambda^* = 1$$

Combinación Convexa con Error Cuadrático

- La función a minimizar es

$$\arg\min_{\lambda\in[0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} (\lambda c_i + d_i)^2$$

- La derivada es

$$\mathcal{J}'(\lambda) = \sum_{i=1}^{m} 2c_i(\lambda c_i + d_i)$$

- Como es derivable, resolviendo $\mathcal{J}'(\lambda) = 0$ obtenemos

$$\lambda' = -\frac{\sum_{i=1}^{m} d_i c_i}{\sum_{i=1}^{m} (c_i)^2}$$

- La solución es entonces $\lambda^* = \max(\min(\lambda', 1), 0)$

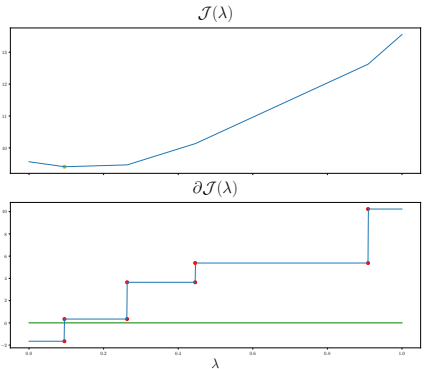
Combinación Convexa con Error Absoluto

- Hay que resolver

$$\arg\min_{\lambda\in[0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} |\lambda c_i + d_i|$$

- El subgradiente de cada sumando es

$$\partial\,|\lambda c_i + d_i| = \begin{cases} -|c_i|\,, & \lambda c_i + d_i < 0, \\ [-|c_i|\,, |c_i|], & \lambda c_i + d_i = 0, \\ |c_i|\,, & \lambda c_i + d_i > 0 \end{cases}$$



Combinación Convexa con Error Absoluto

Proposicion (λ^* óptimo para el problema con valor absoluto)

- $\lambda^* = 0$ *es óptimo si y solo si:* $-\sum_{i:\ 0>\lambda_{(i)}} \left|c_{(i)}\right| + \sum_{i:\ 0<\lambda_{(i)}} \left|c_{(i)}\right| \leq 0$
- $\lambda^* \in (0,1)$ *es óptimo si y solo si* $0 < \lambda^* = \lambda_{(k)} < 1$ *para algún* $k = 1, \ldots, m$, *y*

$$-\sum_{i:\ \lambda_{(k)}>\lambda_{(i)}} \left|c_{(i)}\right| + \sum_{i:\ \lambda_{(k)}<\lambda_{(i)}} \left|c_{(i)}\right| \in \left[-\left|c_{(k)}\right|, \left|c_{(k)}\right|\right]$$

- $\lambda^* = 1$ *es óptimo en otro caso*

Combinación Convexa: Tabla

	$\boldsymbol{\lambda^* \in (0,1)}$
Cuadrática	$0 < -\frac{\sum_{i=1}^{m} d_i c_i}{\sum_{i=1}^{m} (c_i)^2} < 1$
Absoluta	$-\sum_{i:\ \lambda_{(k)} > \lambda_{(i)}} \left\lvert c_{(i)} \right\rvert + \sum_{i:\ \lambda_{(k)} < \lambda_{(i)}} \left\lvert c_{(i)} \right\rvert \in \left[-\left\lvert c_{(k)} \right\rvert, \left\lvert c_{(k)} \right\rvert \right]$
Hinge	$-\sum_{i:\ \lambda_{(k)} > \lambda_{(i)}} \text{máx}\left(0, c_{(i)}\right) - \sum_{i:\ \lambda_{(k)} < \lambda_{(i)}} \text{mín}\left(0, c_{(i)}\right) \in \left[\text{mín}\left(0, c_{(k)}\right), \text{máx}\left(0, c_{(k)}\right)\right]$
Hinge Cuad.	$-\frac{\sum_{i:\ \lambda_{(k+1)} \geq \lambda_{(i)}} \text{máx}\left(0, c_{(i)}\right) d_{(i)} + \sum_{i:\ \lambda_{(k)} \leq \lambda_{(i)}} \text{mín}\left(0, c_{(i)}\right) d_{(i)}}{\sum_{i:\ \lambda_{(k+1)} \geq \lambda_{(i)}} \text{máx}\left(0, c_{(i)}\right)^2 + \sum_{i:\ \lambda_{(k)} \leq \lambda_{(i)}} \text{mín}\left(0, c_{(i)}\right)^2} \in [\lambda_{(k)}, \lambda_{(k+1)}]$
	$\boldsymbol{\lambda^* = 0}$
Cuadrática	$-\frac{\sum_{i=1}^{m} d_i c_i}{\sum_{i=1}^{m} (c_i)^2} \leq 0$
Absoluta	$-\sum_{i:\ 0 > \lambda_{(i)}} \left\lvert c_{(i)} \right\rvert + \sum_{i:\ 0 < \lambda_{(i)}} \left\lvert c_{(i)} \right\rvert \leq 0$
Hinge	$-\sum_{i:\ 0 > \lambda_{(i)}} \text{máx}\left(0, c_{(i)}\right) - \sum_{0 < \lambda_{(i)}} \text{mín}\left(0, c_{(i)}\right) \leq 0$
Hinge Cuad.	$-\sum_{i:\ 0 > c_{(i)}, 0 < \lambda_{(i)}} 2 c_i d_i - \sum_{i:\ 0 < c_{(i)}, 0 > \lambda_{(i)}} 2 c_{(i)} d_{(i)} \leq 0$
	$\boldsymbol{\lambda^* = 1}$ **en otro caso**

- Apéndice A: Combinación Convexa de Modelos Preentrenados
- Apéndice B: Aprendizaje MT Convexo con Redes Neuronales
- Apéndice C: Laplaciano de Grafo con Métodos de Kernel
- Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo
- Apéndice E: Extensiones a L2 y LS-SVM

Formulación Convexa para Redes Neuronales MT

- Proponemos[12] la formulación convexa para redes neuronales MT, combinando:
  - Una parte común $g(\cdot; w, b, \Theta)$
  - Una parte específica $g_r(\cdot; w_r, d_r, \Theta_r)$
- Los modelos son:

$$\begin{aligned} h_r(\cdot) &= \lambda_r g(\cdot; w, b, \Theta) + (1 - \lambda_r) g_r(\cdot; w_r, d_r, \Theta_r) \\ &= \lambda_r \{\langle w, f(\cdot; \Theta)\rangle + b\} + (1 - \lambda_r)\{\langle w_r, f_r(\cdot; \Theta_r)\rangle + d_r\} \end{aligned}$$

  - w, Θ son los parámetros de la red común
  - w_r, Θ_r son los parámetros de las redes específicas
- No se comparte la representación, se combinan una parte común y partes específicas

[12] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Multi-Task Learning with Neural Networks". En: *HAIS Proceedings*. Vol. 13469. Springer, 2022, págs. 223-235.

Formulación Convexa para Redes Neuronales MT

8. Apéndice B: Aprendizaje MT Convexo con Redes Neuronales

- El riesgo a minimizar en este caso es

$$\hat{R}_D = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(h_r(x_i^r), y_i^r) + \frac{\mu}{2} \left(\|w\|^2 + \sum_{r=1}^{T} \|w_r\|^2 + \Omega(\Theta) + \Omega(\Theta_r) \right)$$

- Se puede aplicar el descenso por gradiente con

$$\begin{aligned}
&\nabla_w h_t(x_i^t) = \lambda_t f(x_i^t, \Theta), && \nabla_\Theta h_t(x_i^t) = \lambda_t \left\langle w, \nabla_\Theta f(x_i^t, \Theta) \right\rangle; \\
&\nabla_{w_t} h_t(x_i^t) = (1-\lambda_t) f_t(x_i^t, \Theta), && \nabla_{\Theta_t} h_t(x_i^t) = (1-\lambda_t) \left\langle w, \nabla_{\Theta_t} f_t(x_i^t, \Theta_t) \right\rangle; \\
&\nabla_{w_r} h_t(x_i^t) = 0, && \nabla_{\Theta_r} h_t(x_i^t) = 0, \text{ for } r \neq t
\end{aligned}$$

- Los gradientes se escalan adecuadamente con λ_t y $(1-\lambda_t)$

Experimentos: Modelos

- Comparamos cuatro modelos:
  - ctlNN_conv
  - itlNN_conv
  - cvxmtlNN_conv
  - hsmtlNN_conv
- Todos están basados en una red convolucional de `Pytorch` con
  - Conv. Layer (10 output channels)
  - Conv. Layer (20 output channels)
  - Dropout ($p = 0{,}5$) and Max. Pooling
  - Fully Connected Layer (320 neurons)
  - Fully Connected Layer (50 neurons)

► Apéndice A: Combinación Convexa de Modelos Preentrenados

► Apéndice B: Aprendizaje MT Convexo con Redes Neuronales

► Apéndice C: Laplaciano de Grafo con Métodos de Kernel

► Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo

► Apéndice E: Extensiones a L2 y LS-SVM

Aprendizaje Multitarea con Regularización Laplaciana

9. Apéndice C: Laplaciano de Grafo con Métodos de Kernel

- Dados los modelos para cada tarea definidos como

$$h_r(\cdot) = \langle w_r, \cdot \rangle + b_r,$$

definimos la regularización

$$\sum_{r=1}^{T} \sum_{s=1}^{T} (A)_{rs} \left\| w_r - w_s \right\|^2$$

- Esta regularización se puede expresar como

$$\sum_{r=1}^{T} \sum_{s=1}^{T} (A)_{rs} \left\| w_r - w_s \right\|^2 = \sum_{r=1}^{T} \sum_{s=1}^{T} (L)_{rs} \langle w_r, w_s \rangle$$

- Proponemos combinar la formulación convexa con la regularización Laplaciana[13]

[13] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Convex Graph Laplacian Multi-Task Learning SVM". En: *ICANN*. Vol. 12397. Springer, 2020, págs. 142-154.

Laplaciano de Grafo con Métodos de Kernel

- Inspirados por trabajos previos[14] proponemos lo siguiente:
  - Consideramos el problema de minimización (con E una matriz def. pos.)

$$R(u_1, \ldots, u_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle u_r, \phi(x_i^r) \rangle) + \mu \sum_{r} \sum_{s} (E)_{rs} \langle u_r, u_s \rangle \tag{1}$$

  - Si usamos el vector $\boldsymbol{u}^\intercal = (u_1^\intercal, \ldots, u_T^\intercal)$ lo expresamos como

$$R(\boldsymbol{u}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle \boldsymbol{u}, e_r \otimes \phi(x_i^r) \rangle) + \mu \left(\boldsymbol{u}^\intercal (E \otimes I) \boldsymbol{u} \right) \tag{2}$$

    donde $\otimes$ indica el producto tensorial y $e_1, \ldots e_T$ es la base canónica de $\mathbb{R}^T$

[14] Theodoros Evgeniou, Charles A. Micchelli y Massimiliano Pontil. "Learning Multiple Tasks with Kernel Methods". En: *J. Mach. Learn. Res.* 6 (2005), págs. 615-637.

Lema

Las soluciones $u_1^*, \ldots, u_T^*$ *de* (1)*, o equivalentemente la solución* $\boldsymbol{u}^*$ *de* (2)*, se pueden obtener minimizando*

$$S(\boldsymbol{w}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle \boldsymbol{w}, (B_r \otimes \phi(x_i^r)) \rangle) + \mu \boldsymbol{w}^\top \boldsymbol{w},$$

donde $\boldsymbol{w} \in \mathbb{R}^p \otimes \mathcal{H}$ *con* $p \geq T$ *y* B_r *son las columnas de* $B \in \mathbb{R}^{p \times T}$*, una matriz de rango máximo tal que* $E^{-1} = B^\top B$*.*

El kernel reproductor correspondiente es:

$$\left\langle B_r \otimes \phi(x_i^r), B_s \otimes \phi(x_j^s) \right\rangle = \left(E^{-1}\right)_{rs} k(x_i^r, x_j^s)$$

► Apéndice A: Combinación Convexa de Modelos Preentrenados

► Apéndice B: Aprendizaje MT Convexo con Redes Neuronales

► Apéndice C: Laplaciano de Grafo con Métodos de Kernel

► Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo

► Apéndice E: Extensiones a L2 y LS-SVM

Algoritmo Adaptativo para Laplaciano de Grafo

10. Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo

- La selección de la matriz de adyacencia A (y la respectiva L) es determinante
- Proponemos[15] un método para la selección automática de A
- Tiene que tener las siguientes restricciones:
  - A es simétrica
  - $(A)_{rs} \geq 0, r, s = 1, \ldots, T.$
  - $\sum_{s=1}(A)_{rs} = 1$
- La entropía de cada fila $\boldsymbol{a}^r$ es: $H(\boldsymbol{a}^r) = \sum_{s=1}^{T}(A)_{rs}\log((A)_{rs})$
- Interpretación:
  - $\sum_{r=1}^{T} H(\boldsymbol{a}^r)$ es máxima si A es constante, $A = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\intercal$
  - $\sum_{r=1}^{T} H(\boldsymbol{a}^r)$ es mínima si A es la identidad, $A = I_T$

[15] Carlos Ruiz, Carlos M. Alaíz y José R. Dorronsoro. "Adaptive Graph Laplacian for Convex Multi-Task Learning SVM". En: *HAIS Proceedings*. Vol. 12886. Springer, 2021, págs. 219-230.

Algoritmo Adaptativo para Laplaciano de Grafo

Problema para Algoritmo Adaptativo

$$\begin{aligned}\min_{\substack{\boldsymbol{w},\boldsymbol{v},\boldsymbol{b};\\ A\in(\mathbb{R}_{\geq 0})^{T\times T},\\ A\mathbf{1}_T=\mathbf{1}_T}} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(\lambda_r\langle w,\phi(x_i^r)\rangle+(1-\lambda_r)\langle v_r,\psi(x_i^r)\rangle+b_r,y_i^r)\\ & +\frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r-v_s\|^2+\frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2+\frac{1}{2}\|w\|^2\\ & -\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r)\end{aligned}$$

Algoritmo Adaptativo para Laplaciano de Grafo

10. Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo

- Para minimizar este problema alternamos los siguientes pasos:
  - Fijamos A y minimizamos en $w, \boldsymbol{v}, \boldsymbol{b}$:

$$\min_{w,\boldsymbol{v},\boldsymbol{b}} \quad C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r \langle w, \phi(x_i^r) \rangle + (1-\lambda_r) \langle v_r, \psi(x_i^r) \rangle + b_r, y_i^r) + \frac{\nu}{2} \sum_{r=1}^{T} \sum_{s=1}^{T} (A)_{rs} \|v_r - v_s\|^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2$$

  - Fjamos $w, \boldsymbol{v}, \boldsymbol{b}$ y minimizamos en A:

$$\min_{\substack{A \in (\mathbb{R}_{\geq 0})^{T \times T}, \\ A\mathbf{1}_T = \mathbf{1}_T}} J(A) = \frac{\nu}{2} \sum_{r=1}^{T} \sum_{s=1}^{T} (A)_{rs} \|v_r - v_s\|^2 - \mu \sum_{r=1}^{T} H(\boldsymbol{a}^r)$$

Algoritmo Adaptativo para Laplaciano de Grafo

10. Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo

- Si sabemos las distancias $\|v_r - v_s\|^2$, la solución es

$$(A)_{rs} = \frac{\exp -\frac{\nu}{\mu}\|v_r - v_s\|^2}{\sum_t \exp -\frac{\nu}{\mu}\|v_r - v_t\|^2}$$

- ¿Cómo calculamos estas distancias?
  - Con la matriz $\widetilde{Q^{rs}}$ correspondiente a la función

$$\widetilde{k^{rs}}(x_i^t, x_j^\tau) = (I_T + \nu L)^{-1}_{rt}(I_T + \nu L)^{-1}_{s\tau} k_\psi(x_i^t, x_j^\tau)$$

  - Los productos interiores son

$$\langle v_r, v_s \rangle = \boldsymbol{\alpha}^\intercal \left(I_n - \Lambda\right) \widetilde{Q^{rs}} \left(I_n - \Lambda\right) \boldsymbol{\alpha}$$

  - Las distancias son entonces

$$\|v_r - v_s\|^2 = \boldsymbol{\alpha}^\intercal \left(I_n - \Lambda\right) \left(\widetilde{Q^{rr}} + \widetilde{Q^{ss}} - 2\widetilde{Q^{rs}}\right) \left(I_n - \Lambda\right) \boldsymbol{\alpha}$$

Algoritmo 1: Algoritmo para laplaciano adaptativo.

Input: $(X, y) = \{(x_i^r, y_i^r), i = 1, \ldots, m_r; r = 1, \ldots, T\}$ `// Data`

$A = A_0$ `// Constant matrix`

while *True* **do**

  $L_{\text{inv}} \leftarrow$ getInvLaplacian(A) `// Step 0`

  $\alpha_{\text{opt}} \leftarrow$ solveDualProblem((X, y), L_{inv}, params) `// Step 1`

  $o \leftarrow$ computeObjectiveValue((X, y), L_{inv}, α_{opt}) `// Objective function value`

  if $o^{old} - o \leq \delta_{tol}$ **then**

    break `// Exit condition`

  end

  $D \leftarrow$ computeDistances((X, y), L_{inv}, α_{opt}) `// Step 2`

  $A \leftarrow$ updateAdjMatrix(D, params) `// Step 3`

end

return α_{opt}, A

- ▶ Apéndice A: Combinación Convexa de Modelos Preentrenados
- ▶ Apéndice B: Aprendizaje MT Convexo con Redes Neuronales
- ▶ Apéndice C: Laplaciano de Grafo con Métodos de Kernel
- ▶ Apéndice D: Algoritmo Adaptativo para Laplaciano de Grafo
- ▶ Apéndice E: Extensiones a L2 y LS-SVM

L2-SVM

Problema Primal - L2-SVM

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{C}{2}\sum_{i=1}^{m}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq p_i - \xi_i \end{aligned}$$

Problema Dual - L2-SVM

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^\intercal \left(Q + \frac{1}{C}I_n\right)\alpha - \alpha^\intercal \boldsymbol{p} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i\alpha_i = 0,\ \alpha_i \geq 0 \end{aligned}$$

LS-SVM

Problema Primal - LS-SVM

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) = p_i - \xi_i \end{aligned}$$

Problema Dual - LS-SVM

$$\left[\begin{array}{c|c} 0 & \boldsymbol{\gamma}^\intercal \\ \hline \boldsymbol{\gamma} & Q + \frac{1}{C}I_n \end{array}\right] \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix}$$

Formulación Convexa para L2-SVM MT

Problema Primal - L2-SVM MT Convexa

$$\underset{w,\boldsymbol{v},b,\boldsymbol{d},\xi}{\arg\min} \quad \frac{C}{2}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r\left(\lambda_r\left\{\langle w,\phi(x_i^r)\rangle + b\right\} + (1-\lambda_r)\left\{\langle v_r,\phi_r(x_i^r)\rangle + d_r\right\}\right) \geq p_i^r - \xi_i^r$$

Problema Dual - L2-SVM MT Convexa

$$\underset{\alpha}{\min} \quad \frac{1}{2}\boldsymbol{\alpha}^\intercal\left(\left\{\Lambda Q\Lambda + (I_n-\Lambda)K(I_n-\Lambda)\right\} + \frac{1}{C}I\right)\boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha}$$

$$\text{s.t.} \quad 0 \leq \alpha_i^r,\ i=1,\ldots,m_r,\ r=1,\ldots,T,$$

$$\sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0,\ r=1,\ldots,T$$

Formulación Convexa para LS-SVM MT

Problema Primal - LS-SVM MT Convexa

$$\underset{w,\boldsymbol{v},b,\boldsymbol{d},\xi}{\arg\min} \quad \frac{C}{2}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r\left(\lambda_r\left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1-\lambda_r)\left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}\right) = p_i^r - \xi_i^r$$

Problema Dual - LS-SVM MT Convexa

$$\left[\begin{array}{c|c|c} 0 & \mathbf{0}_T^\intercal & \boldsymbol{y}^\intercal\Lambda \\ \hline \mathbf{0}_T & \mathbf{0}_{T\times T} & A^\intercal Y(I_n-\Lambda) \\ \hline \boldsymbol{y} & YA & \left\{\Lambda Q\Lambda + (I_n-\Lambda)K(I_n-\Lambda)\right\} + \frac{1}{C}I \end{array}\right] \begin{pmatrix} b \\ d_1 \\ \vdots \\ d_T \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{0}_T \\ \boldsymbol{p} \end{pmatrix}$$

Formulación Convexa para L2-SVM MT con Laplaciano

Problema Primal - L2-SVM Convexa con Laplaciano

$$\min_{\substack{v_1,\ldots,v_T;\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r - v_s\|^2 + \frac{1}{2}\sum_{r}\|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r(\lambda_r(\langle w, \phi(x_i^r)\rangle) + (1-\lambda_r)(\langle v_r, \psi(x_i^r)\rangle) + b_r) \geq p_i^r - \xi_i^r$$

Problema Dual - L2-SVM Convexa con Laplaciano

$$\min_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t\left\{\left(\Lambda Q\Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda)\right) + \frac{1}{C}I_n\right\}\boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha}$$

$$\text{s.t.} \quad 0 \leq \alpha_i^r,\ i = 1,\ldots,m_r,\ r = 1,\ldots,T,\ \sum_{i=1}^{n_r}\alpha_i^r y_i^r = 0,\ r = 1,\ldots,T$$

Formulación Convexa para LS-SVM MT con Laplaciano

Problema Primal - LS-SVM Convexa con Laplaciano

$$\min_{\substack{v_1,\dots,v_T;\\ b_1,\dots,b_T;\\ \boldsymbol{\xi},w;}} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r - v_s\|^2 + \frac{1}{2}\sum_{r}\|v_r\|^2 + \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i^r(\lambda_r(\langle w, \phi(x_i^r)\rangle) + (1-\lambda_r)(\langle v_r, \psi(x_i^r)\rangle) + b_r) = p_i^r - \xi_i^r$$

Problema Dual - LS-SVM Convexa con Laplaciano

$$\left[\begin{array}{c|c} \mathbf{0}_{T\times T} & A^\intercal Y \\ \hline YA & \left(\Lambda Q\Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda)\right) + \frac{1}{C}I_n \end{array}\right] \begin{bmatrix} b_1 \\ \vdots \\ b_T \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_T \\ \boldsymbol{p} \end{bmatrix}$$