

Tópicos Avanzados en Entrenamiento de Redes Profundas

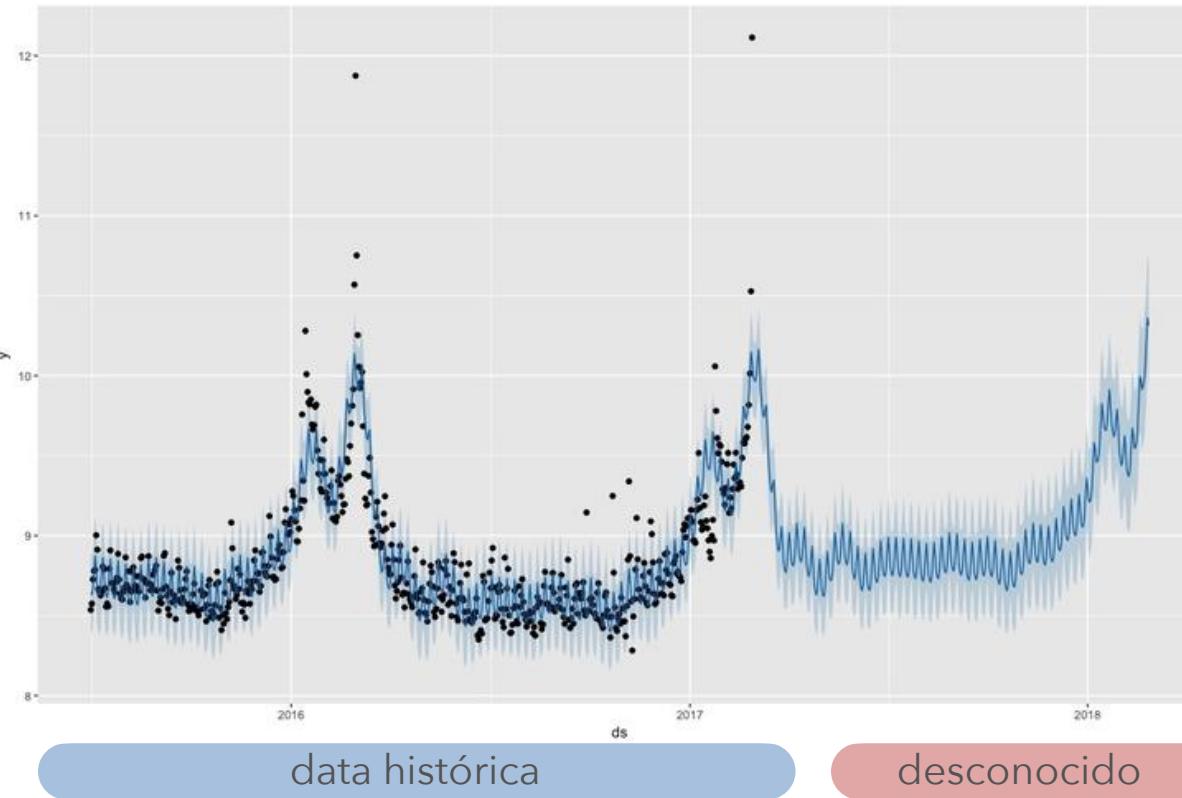
Métodos Clásicos de Regularización



Prof. Ricardo Ñanculef - Departamento de Informática UTFSM

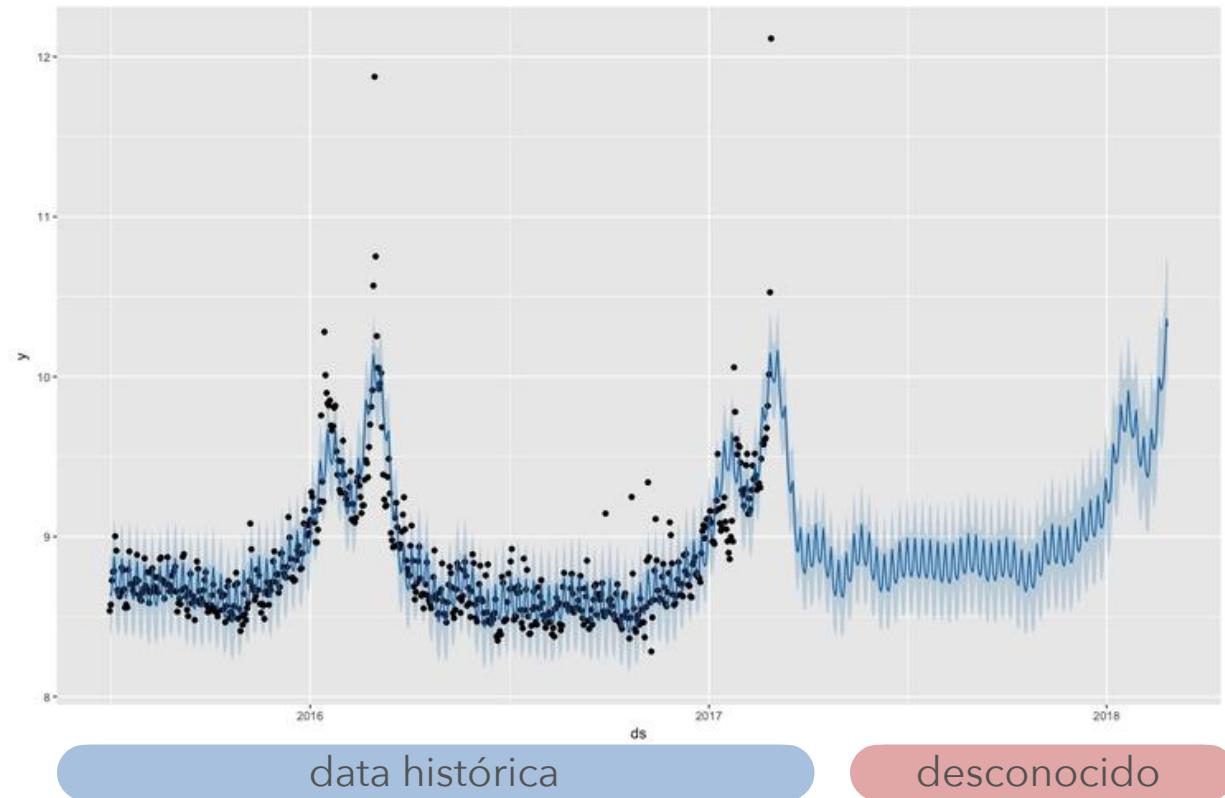
Introducción

- Hemos dicho que (la mayoría de las veces) el objetivo al entrenar nuestra red es maximizar su desempeño sobre casos nuevos o futuros, no vistos exactamente en el conjunto de entrenamiento.



Introducción

- Por ejemplo: si entramos una red para predecir niveles de contaminación en Santiago usando datos históricos, lo que nos interesa realmente es que la red tenga un buen desempeño al predecir los días aún por venir, no aquellos que ya pasaron, aún si son estos últimos los casos con que podemos entrenar.



Introducción

- Más formalmente, si el ambiente genera datos $z \sim P(z)$ y podemos definir una función de costo (loss) $L(f, z)$ asociada a las predicciones de la red f para un determinado caso z , nos interesa minimizar el error de predicción:

$$\mathbb{E}_z(L(f, z))$$

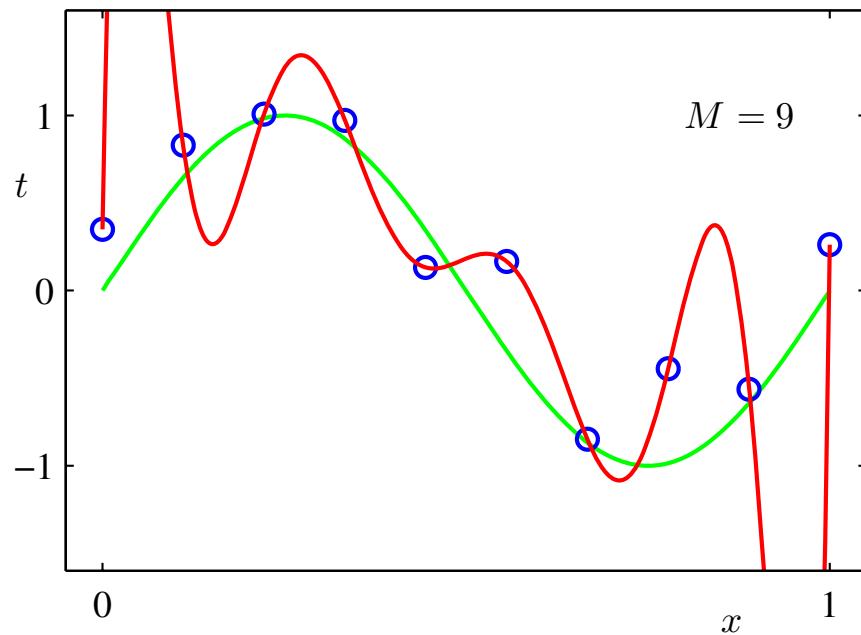
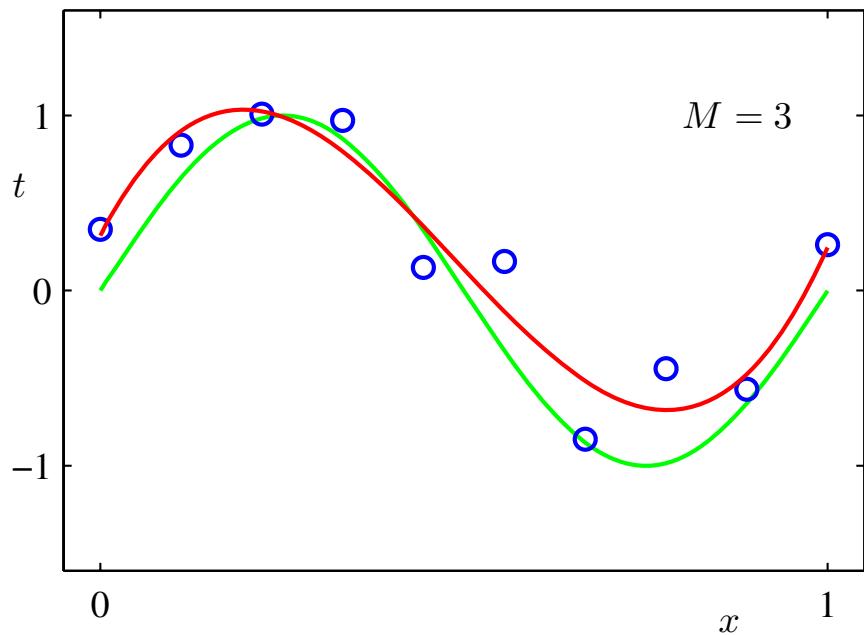
- Típicamente $z = (x, y)$ y $L(f, z) = L(f(x), y)$.
- Sin embargo, durante el entrenamiento, optimizamos la red para minimizar el error observado sobre los ejemplos de entrenamiento:

$$\frac{1}{n} \sum_i L(f_i, z_i) = \frac{1}{n} \sum_i L(f(x^{(i)}), y^{(i)})$$



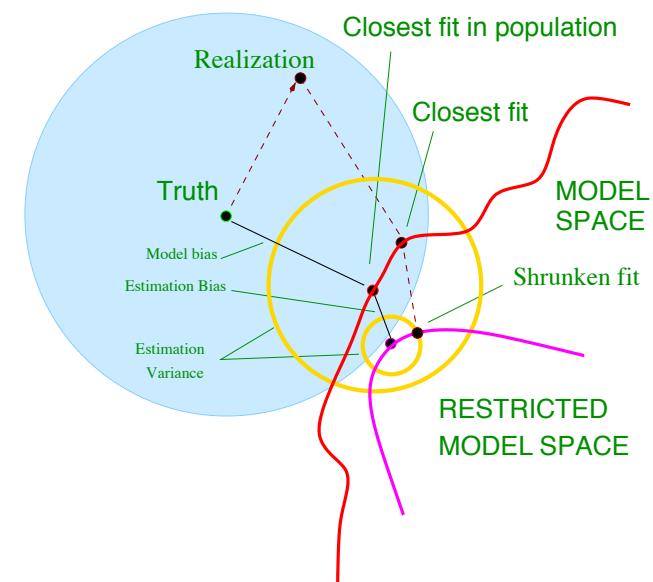
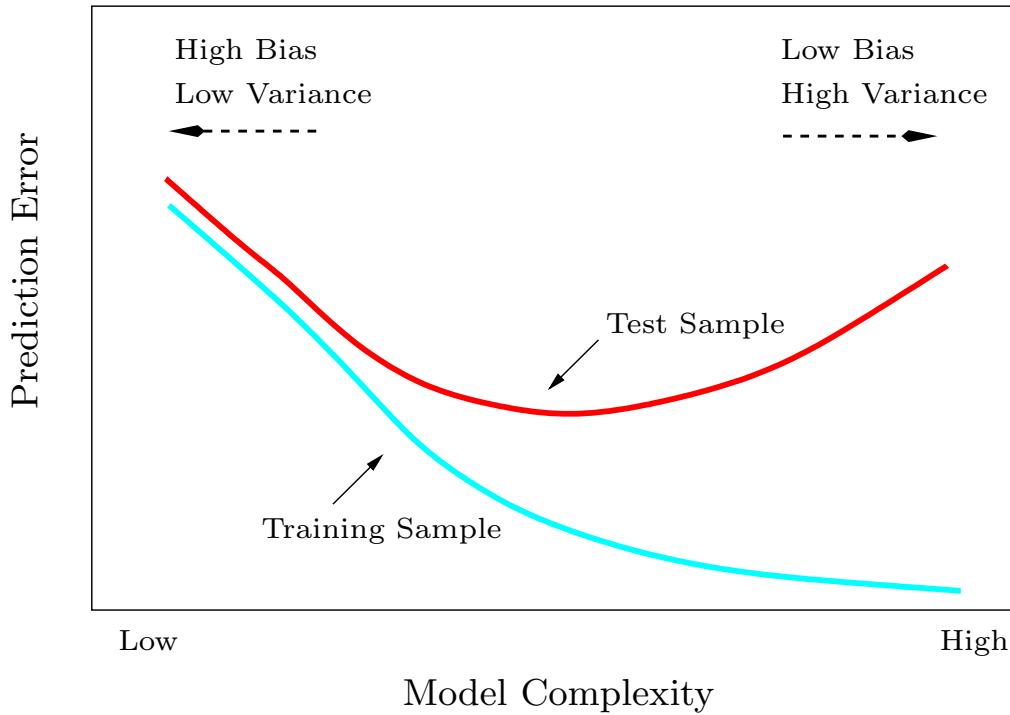
Overfitting

- Lamentablemente, en la práctica, obtener un bajo error de entrenamiento no garantiza un bajo error de predicción.



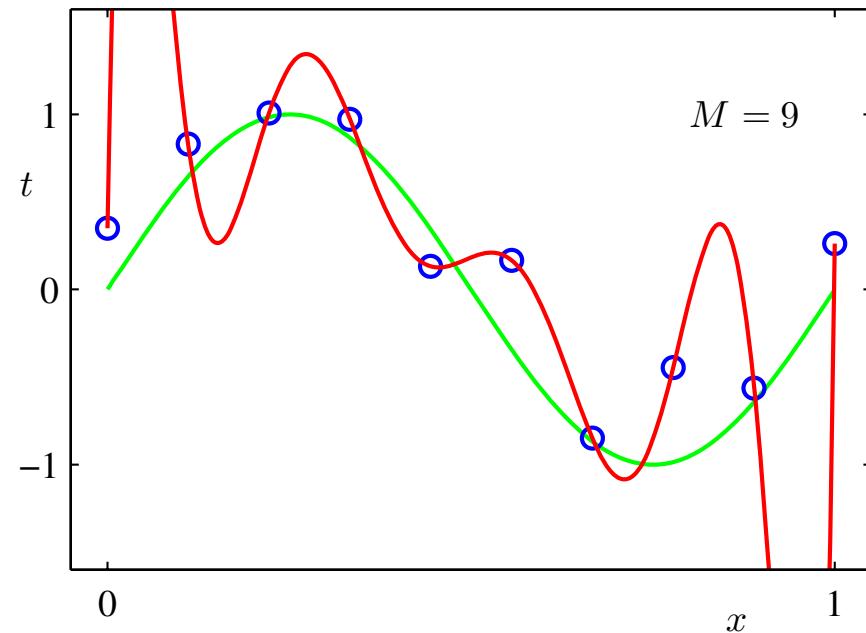
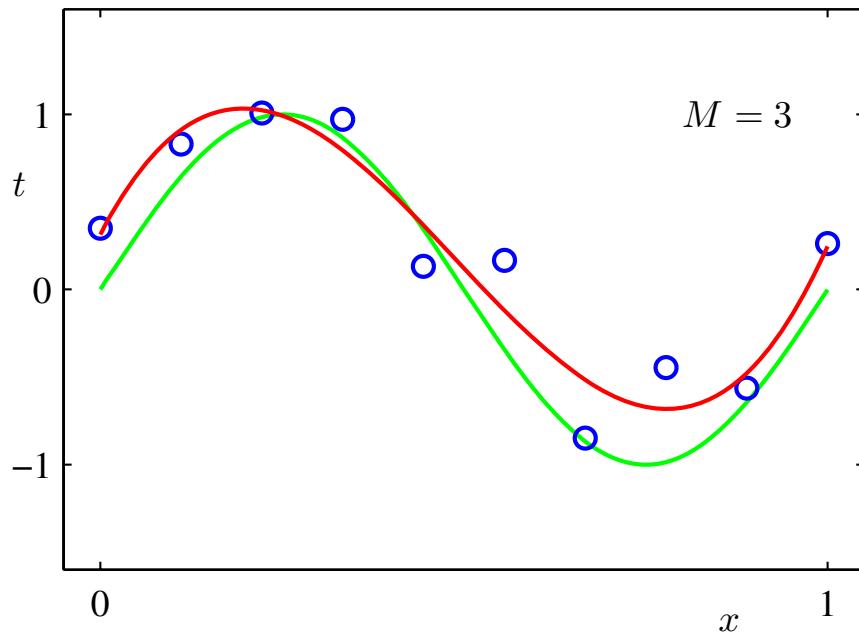
Overfitting

- Esta situación se denomina sobre-ajuste (overfitting) y suele estar asociada ó a una escasez de ejemplos de entrenamiento ó a una capacidad/ complejidad excesiva del espacio de soluciones donde el algoritmo busca una solución.



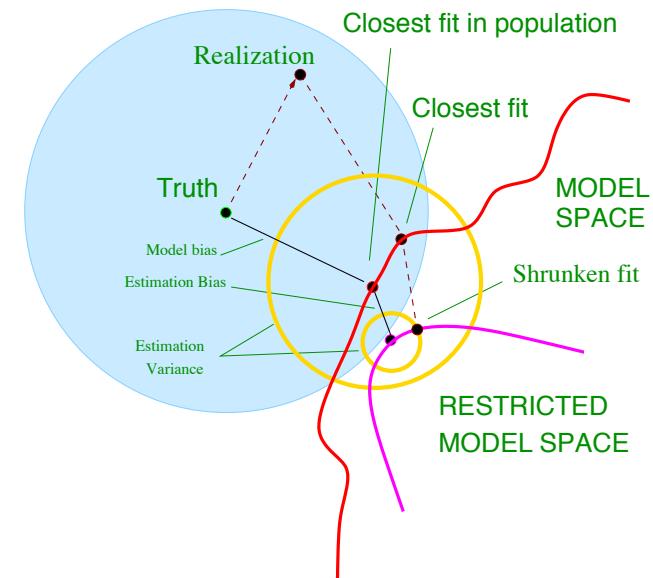
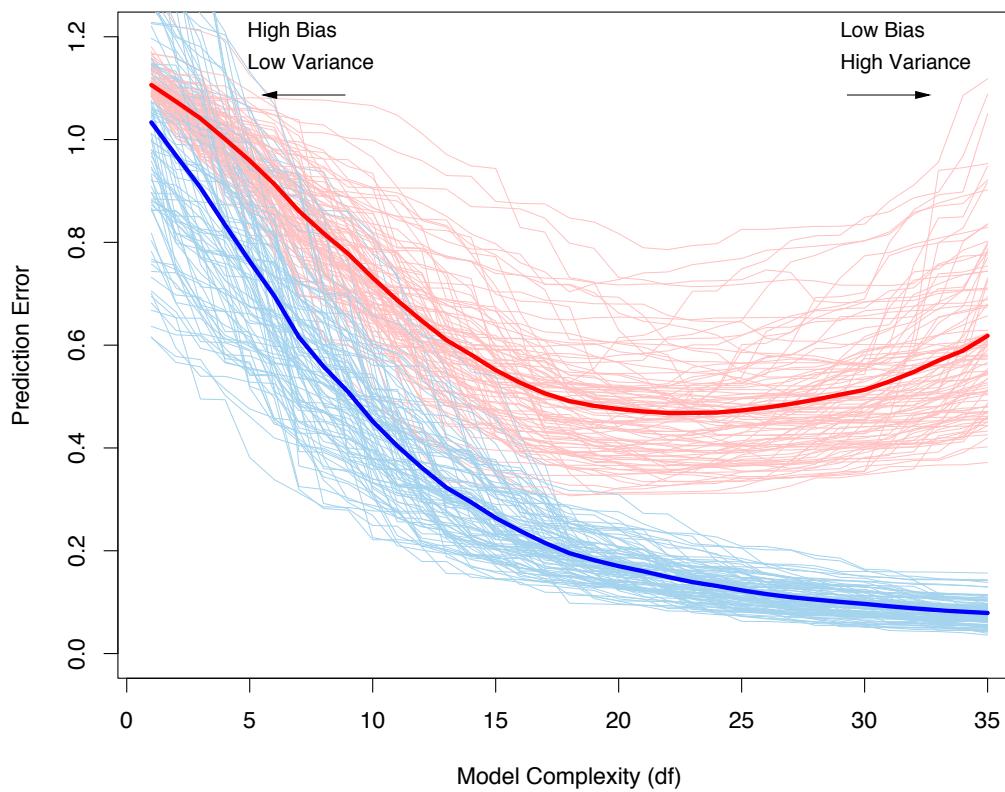
Overfitting vs Capacidad

- Intuitivamente, si el espacio de soluciones es muy flexible, es trivial memorizar los datos de entrenamiento sin necesidad de aprender las regularidades subyacentes.



Overfitting vs Variabilidad

- Intuitivamente, si el espacio de soluciones es muy flexible (ii) la solución obtenida puede variar significativamente aún con un pequeño cambio en el conjunto de entrenamiento.



Sesgo-Varianza

- Con frecuencia se argumenta que la tendencia a overfitting de un modelo se relaciona con una *alta varianza*. Esta idea se puede formalizar al considerar el error cuadrático medio del estimador en la estimación de $y = f(x) + \epsilon$

$$\begin{aligned}\mathbb{E} [(y - f_S(x))^2] &= \mathbb{E} [f_S(x)^2] - 2\mathbb{E} [f_S(x)] \mathbb{E} [y] + \mathbb{E} [y^2] \\ &= \mathbb{V} [f_S(x)] + \mathbb{E} [f_S(x)]^2 - 2\mathbb{E} [f_S(x)] f(x) + \mathbb{V} [y] + \mathbb{E} [y]^2 \\ &= \mathbb{V} [f_S(x)] + \mathbb{E} [f_S(x)]^2 - 2\mathbb{E} [f_S(x)] f(x) + \mathbb{V} [y] + f(x)^2 \\ &= \mathbb{V} [f_S(x)] + (\mathbb{E} [f_S(x)] - f(x))^2 + \mathbb{V} [y] \\ &= \mathbb{V} [f_S(x)] + (\mathbb{E} [f_S(x)] - f(x))^2 + \sigma^2\end{aligned}$$

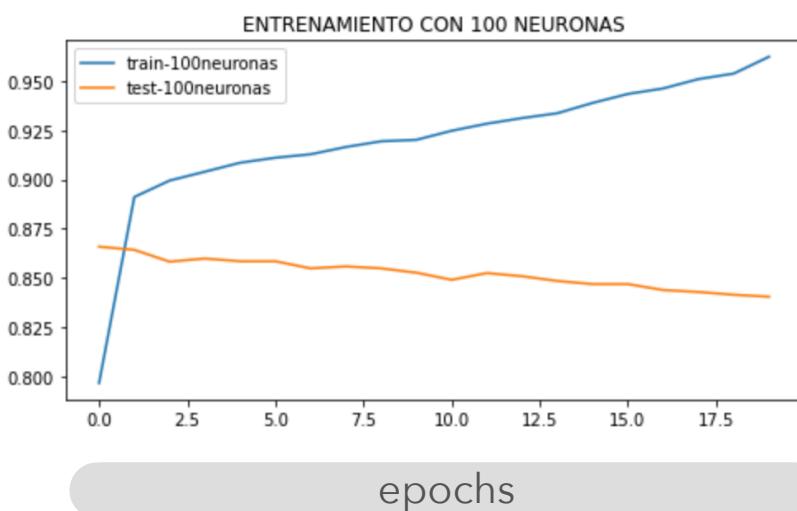
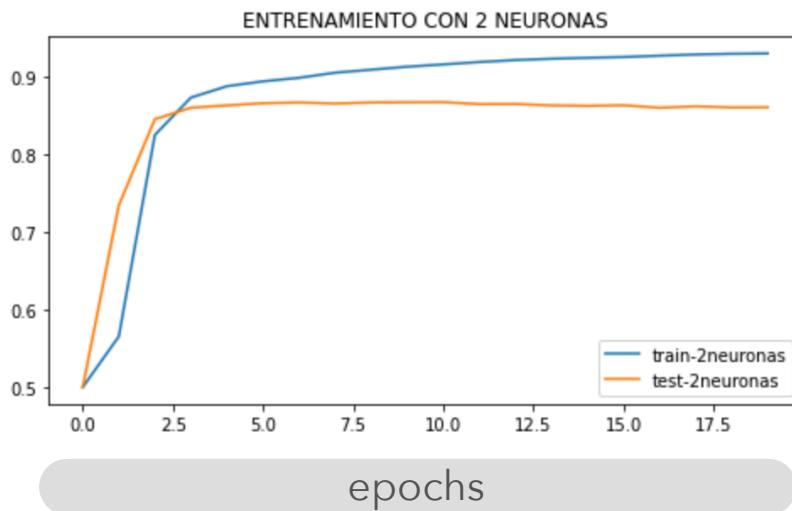
error de predicción = sesgo + varianza + varianza intrínseca de y

- Con una inicialización cuidadosa, las redes obtenidas mediante un agresivo early stopping serán muy similares entre sí i.e. poco variables w.r.t. su valor esperado.



Overfitting en Redes Neuronales

- Como las redes neuronales son aproximadores universales y su expresividad puede aumentar significativamente con la profundidad, no es inusual observar este problema en aplicaciones prácticas.



Entrenamiento, Validación, Test

- El problema de estimar el error de predicción usando el conjunto de entrenamiento es que la red fue **optimizada** para ese conjunto, y por lo tanto no es estadísticamente independiente de esos datos.
- La estrategia más común para estimar el error de predicción consiste en reservar un conjunto de ejemplos que no serán utilizados para entrenar. Este conjunto se denomina **conjunto de pruebas** o test set.



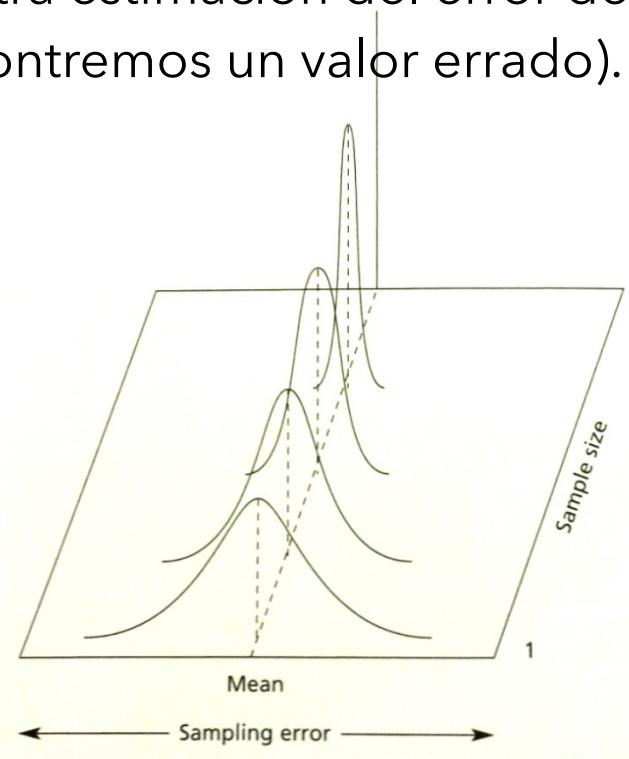
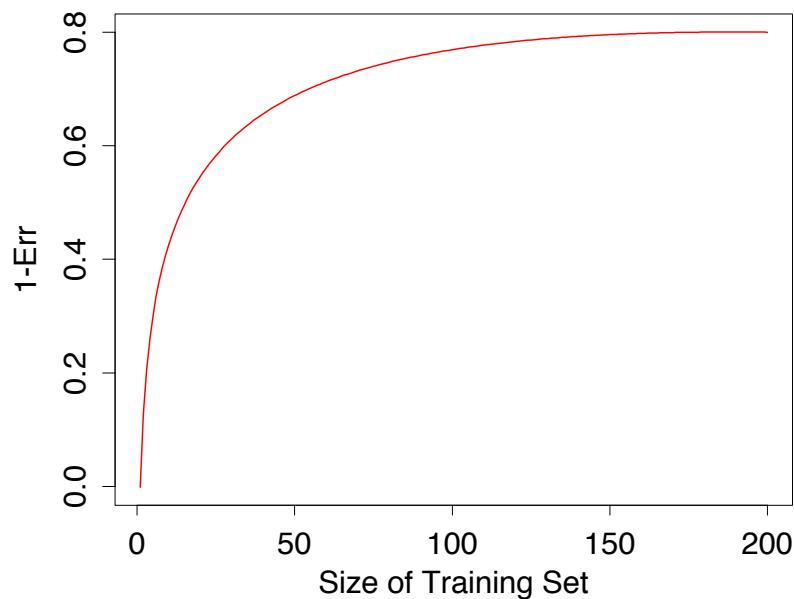
Entrenamiento, Validación, Test

- Es común que *antes de evaluar* el modelo, necesitemos tomar decisiones arquitectónicas como e número de capas y neuronas que conviene utilizar.
- Si usamos el conjunto de pruebas para tomar estas decisiones, caemos en el mismo problema de antes: los datos dejan de ser estadísticamente independientes del modelo porque éste se ha optimizado sobre el.
- La solución más simple es reservar un tercer conjunto para *seleccionar* el modelo correcto. Este conjunto se denomina *conjunto de validación*.



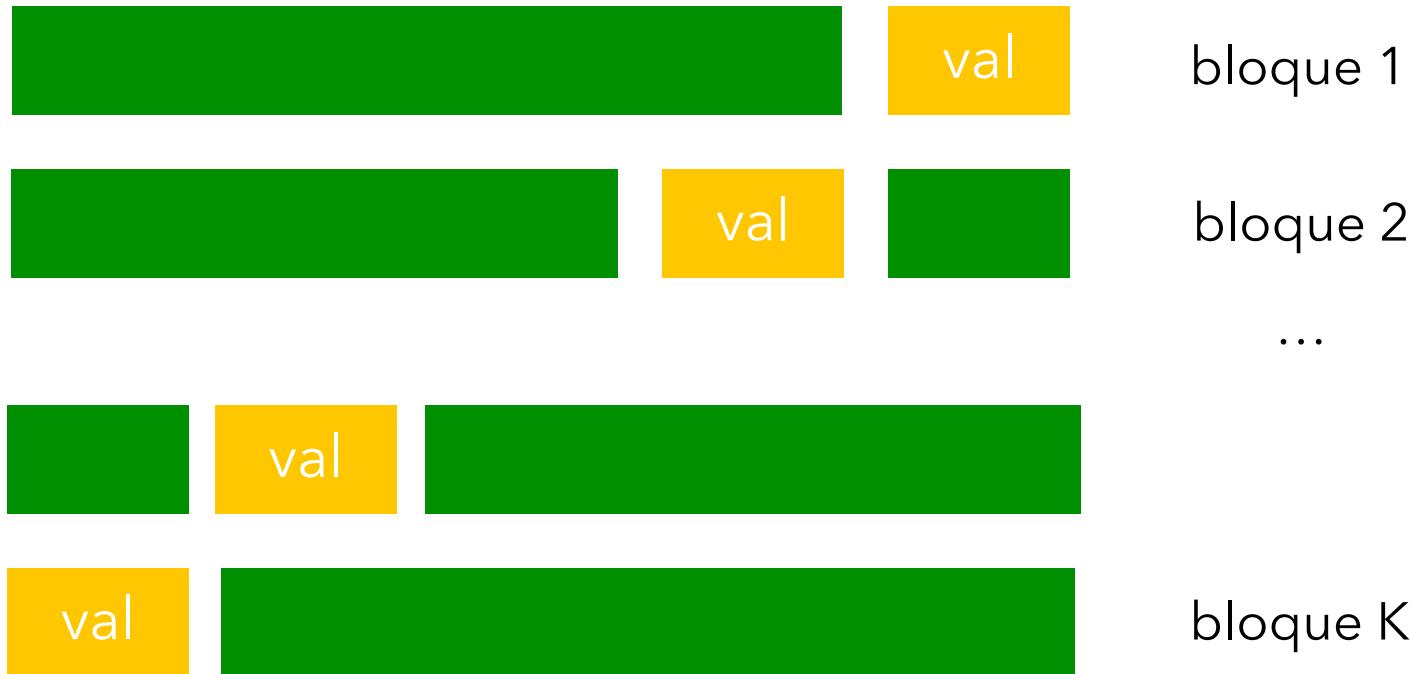
Problema

- Mientras más ejemplos reservemos para validación (selección) o pruebas (evaluación) menos tendremos para entrenar el modelo.
- Mientras menos ejemplos reservemos para validación (selección) o pruebas (evaluación) menos precisa será nuestra estimación del error de predicción (alta varianza, muy probable que encontremos un valor errado).



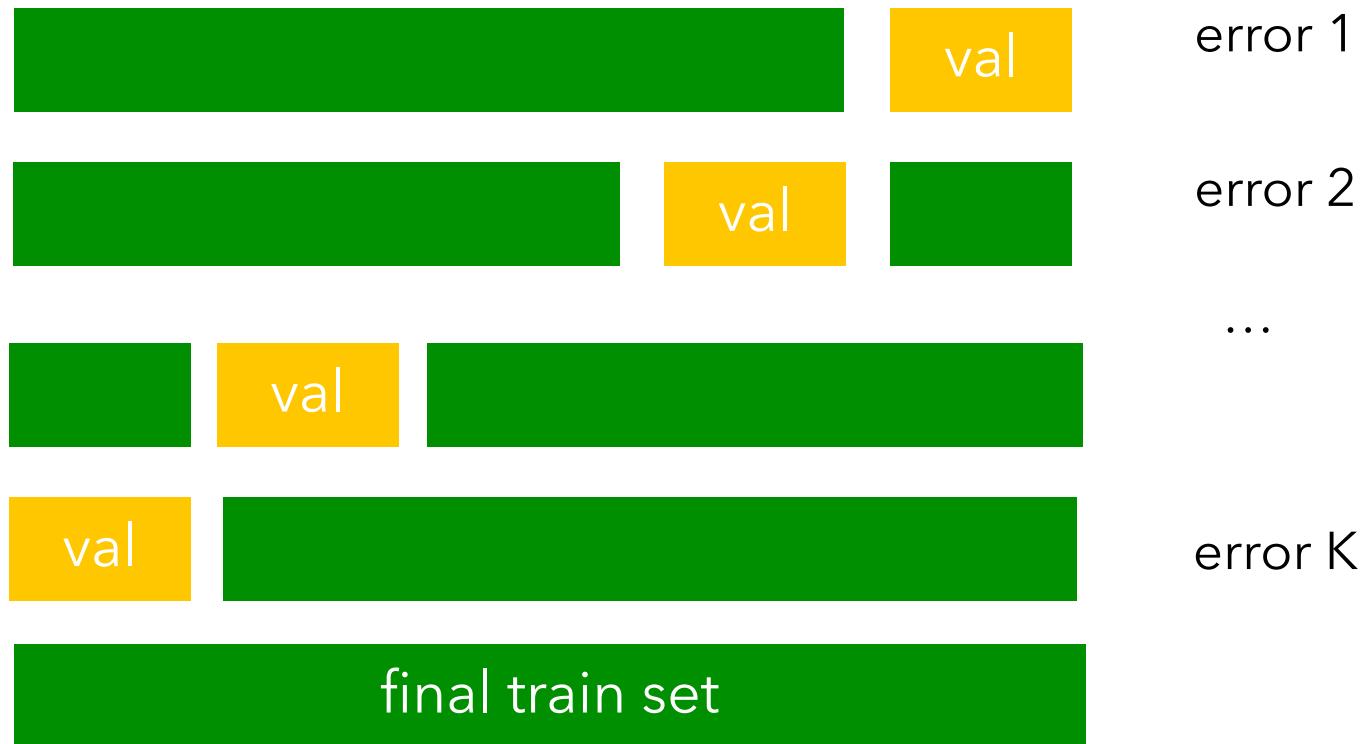
Validación Cruzada

- Una de las soluciones más comunes en machine learning consiste en usar un procedimiento denominado **validación cruzada (K-fold CV)**: el proceso de validación se repite K veces usando en cada ronda un subconjunto diferente de tamaño n/K .



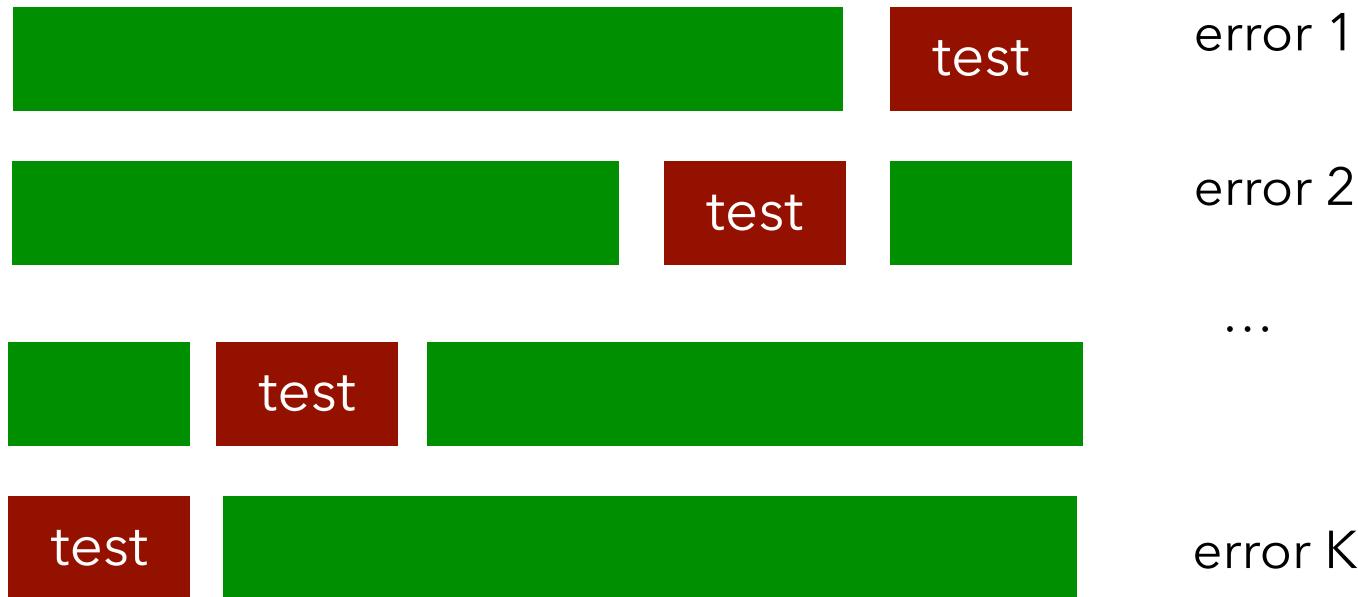
Validación Cruzada

- El error de predicción se estima como el promedio de los errores de predicción en los K bloques de datos.
- Una vez tomadas las decisiones arquitecturales necesarias (e.g. número de capas) el modelo se entrena sobre todo el conjunto de datos.



Validación Cruzada

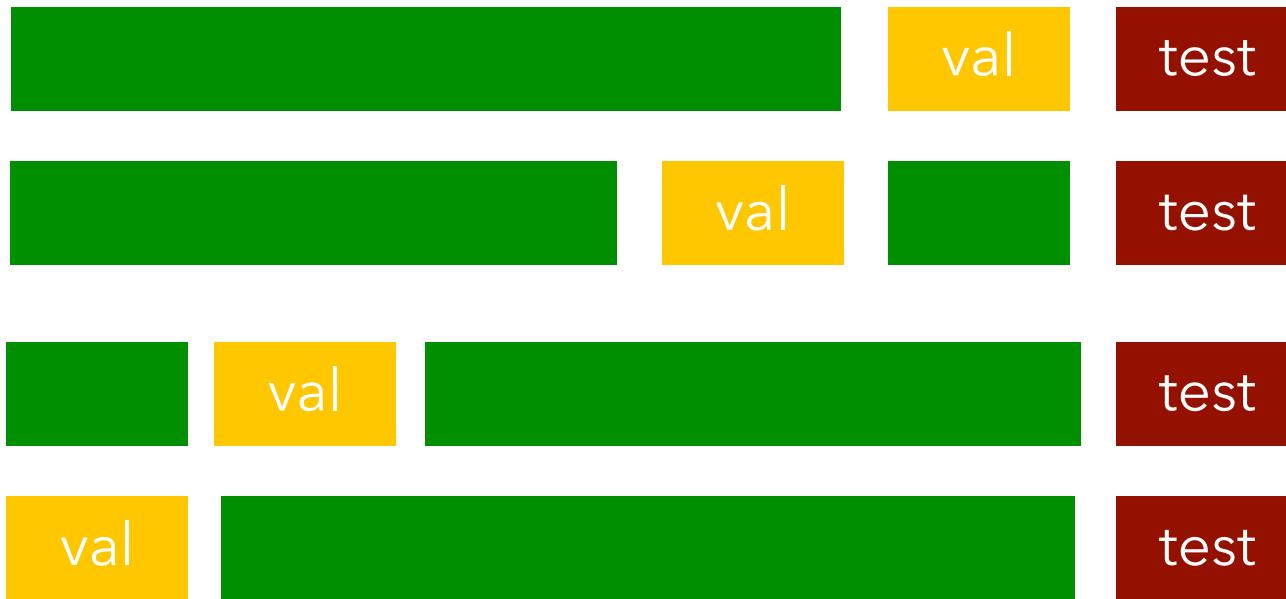
- K-fold CV es un método para estimar el error de predicción. Por lo tanto, es posible también usarlo para *evaluar* el modelo final.



- Sin embargo, en este caso **no** es correcto usar error en los bloques de pruebas para seleccionar el modelo.

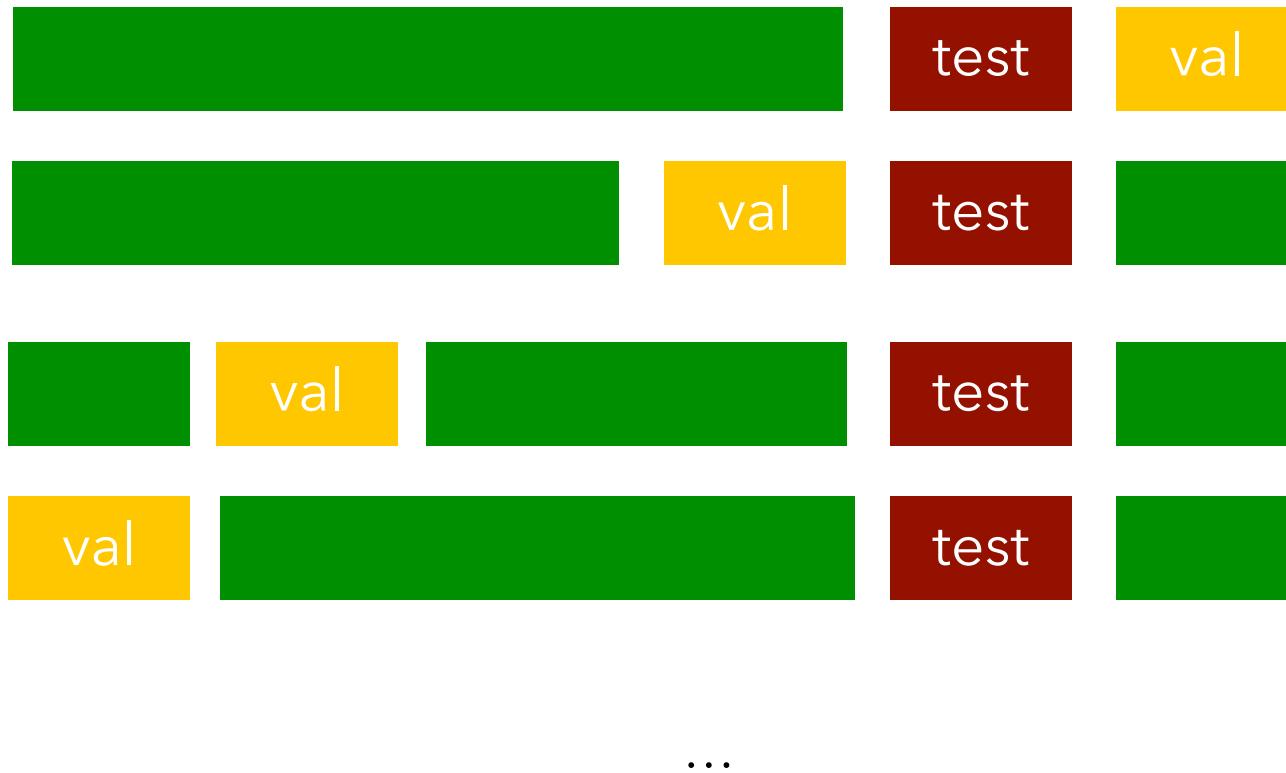
Validación Cruzada

- Si se quiere usar CV tanto para evaluar como para seleccionar el modelo, lo aconsejable es usar una forma anidada de K-fold CV.



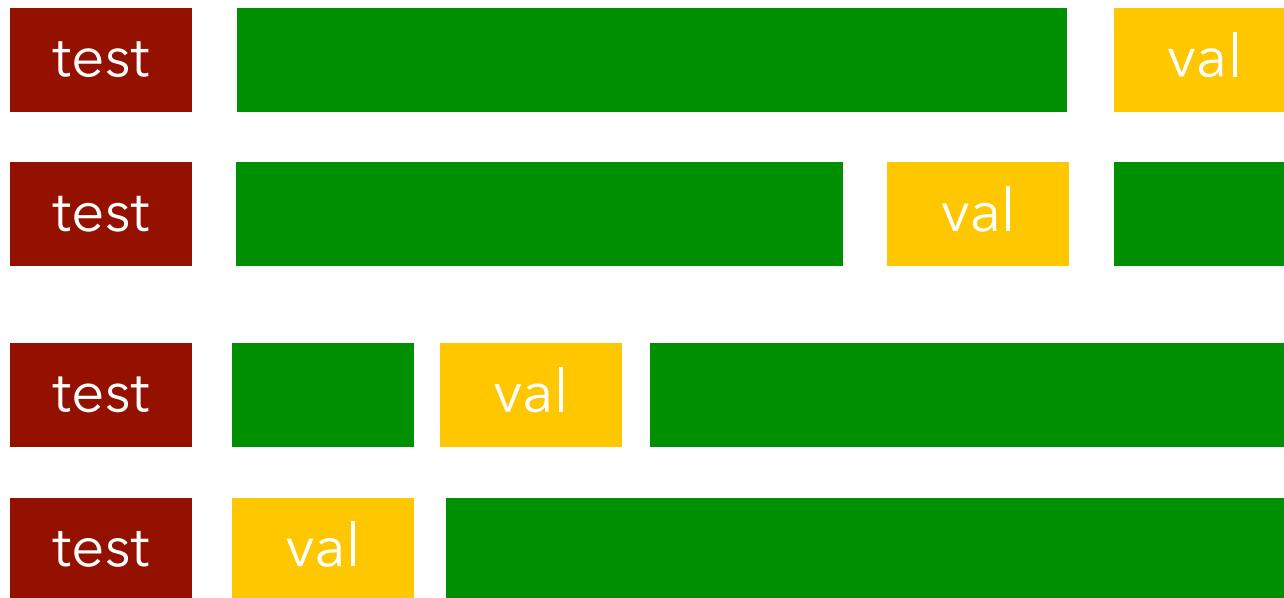
Validación Cruzada

- Si se quiere usar CV tanto para evaluar como para seleccionar el modelo, lo aconsejable es usar una forma anidada de K-fold CV.



Validación Cruzada

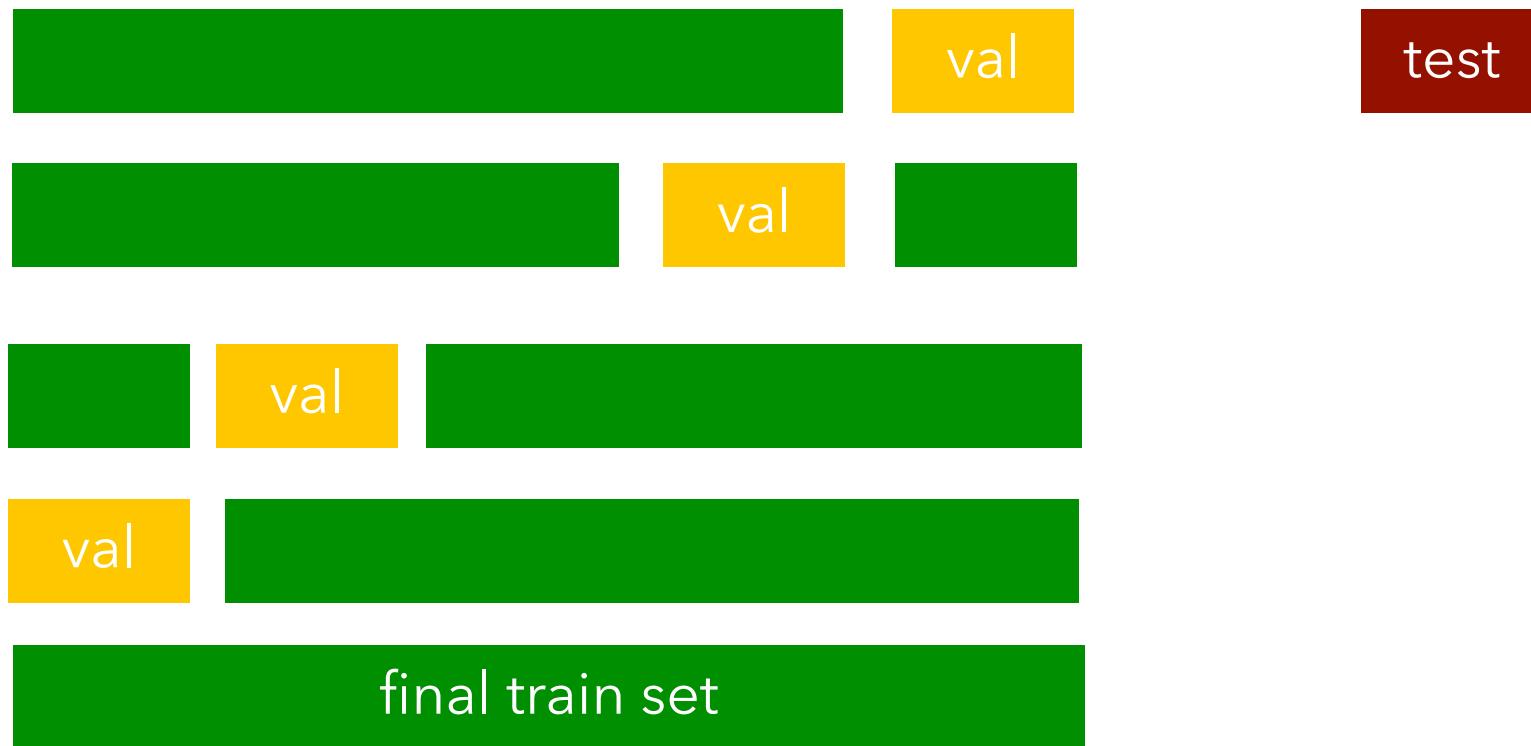
- ...



- Cada vez que fijamos el bloque de pruebas, usamos CV dentro del resto de los datos para tomar decisiones arquitectónicas y finalmente medimos el error de pruebas en ese bloque, repitiendo el proceso K veces.

Validación Cruzada

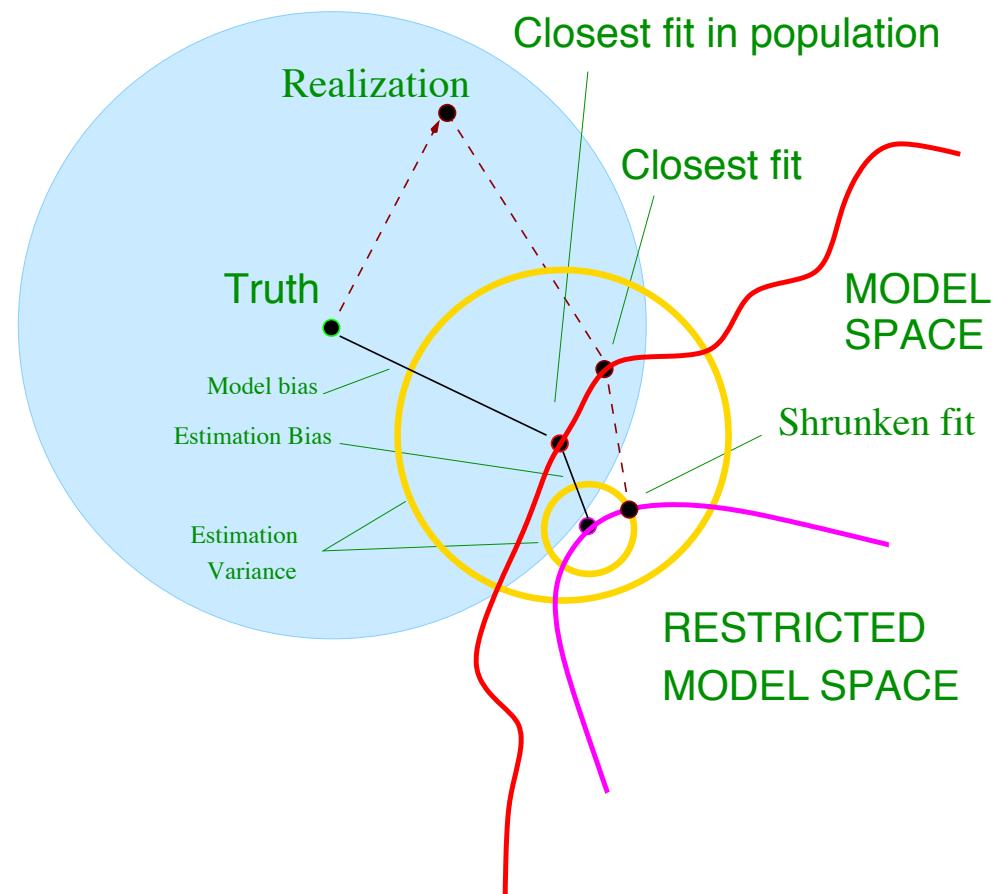
- Debido al alto costo de este procedimiento, en el caso de redes neuronales es usual trabajar con un conjunto fijo de pruebas y validación, o limitar el uso de K-fold CV al proceso de validación (selección) de modelo.



Regularización

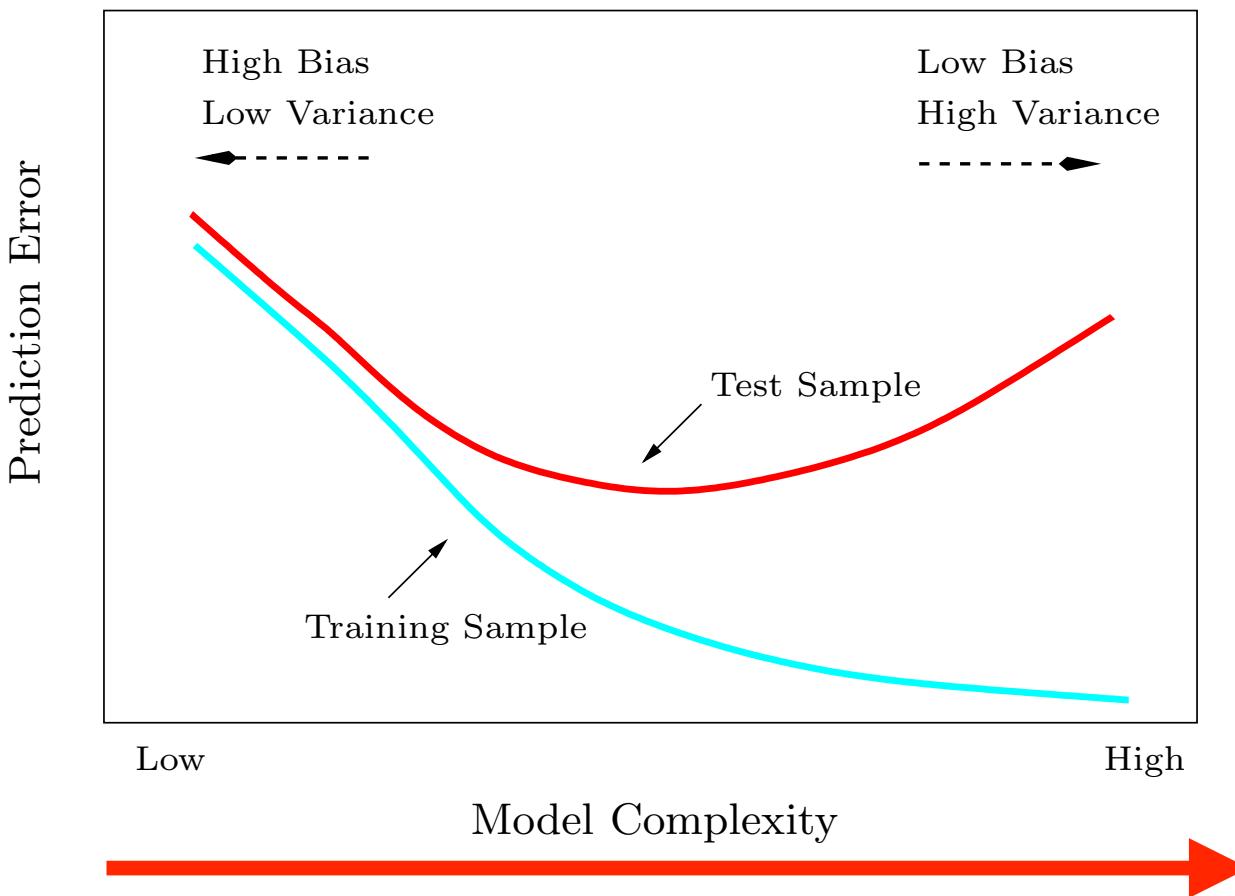
- Los métodos para prevenir o atenuar overfitting se denominan **métodos de regularización.**

- La gran mayoría de estos métodos clásicos trabajan limitando explícita o implícitamente el tamaño del espacio de soluciones que el método puede explorar.



Regularización

- Esquemáticamente, la idea consiste en generar una sucesión creciente (o decreciente) de restricciones del espacio total de soluciones y estimar el error de predicción en cada paso (e.g. usando un conjunto de validación).



Regularización como Prior

- Ese proceso de permisividad creciente de la capacidad del espacio de soluciones parte desde algún punto. Ese punto representa típicamente la solución que nos esperamos obtener a no ser que los datos impongan la necesidad de una modificación a ese modelo de base.
- Es común referirse a este punto como el *prior* del modelo, explotando la interpretación Bayesiana del aprendizaje, que consiste en estimar una distribución de probabilidad condicional sobre el espacio de modelos (o parámetros) dados los datos $p(h | D)$. Por Bayes, tenemos

$$p(h|D) = \frac{p(D|h)p(h)}{p(D)}$$

$$\text{a-posteriori} = \frac{\text{verosimilitud} \times \text{a-priori}}{\text{evidencia}}$$



Regularización como Prior

- Una parte importante de los métodos de aprendizaje utilizados en la práctica entran el modelo maximizando una f.o. proporcional a la verosimilitud. Por ejemplo, el entrenamiento con la loss cuadrática en regresión puede motivarse como sigue:

$$y = f(x) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$p(y - f(x)) = p(\epsilon) \propto \exp\left(-\frac{(y - f(x))^2}{2\sigma^2}\right)$$



$$\ell(\theta) = \log P(D|\theta) \propto \frac{1}{2n\sigma^2} \sum_i (y^{(i)} - f(x^{(i)}))^2$$

iid

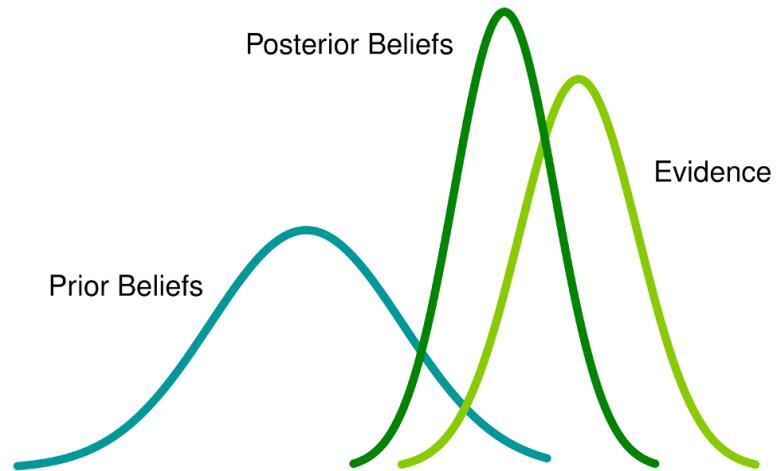
Regularización como Prior

- Imponer un prior permite obtener una solución con características deseadas (simplicidad, suavidad, etc) a no ser que ser consistentes con esta preferencia signifique un gran error de predicción sobre los datos observados.

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

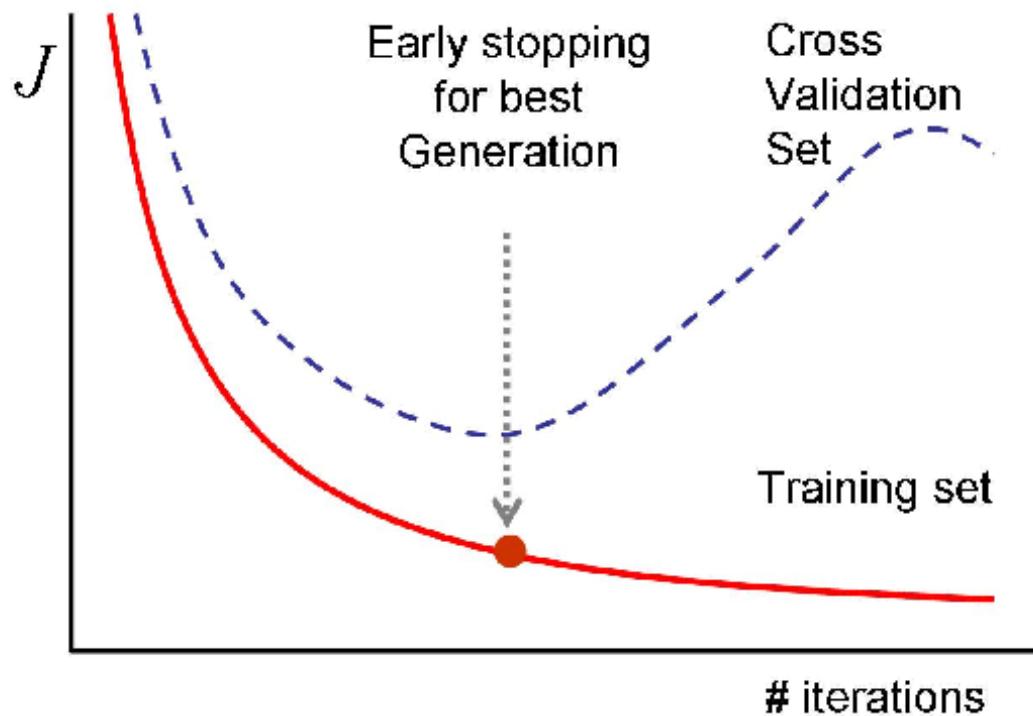
$$\text{a-posteriori} = \frac{\text{verosimilitud} \times \text{a-priori}}{\text{evidencia}} .$$

- La gran mayoría de estos métodos de regularización trabajan imponiendo un prior y permitiendo gradualmente alejarse se éste si el error de predicción lo justifica.



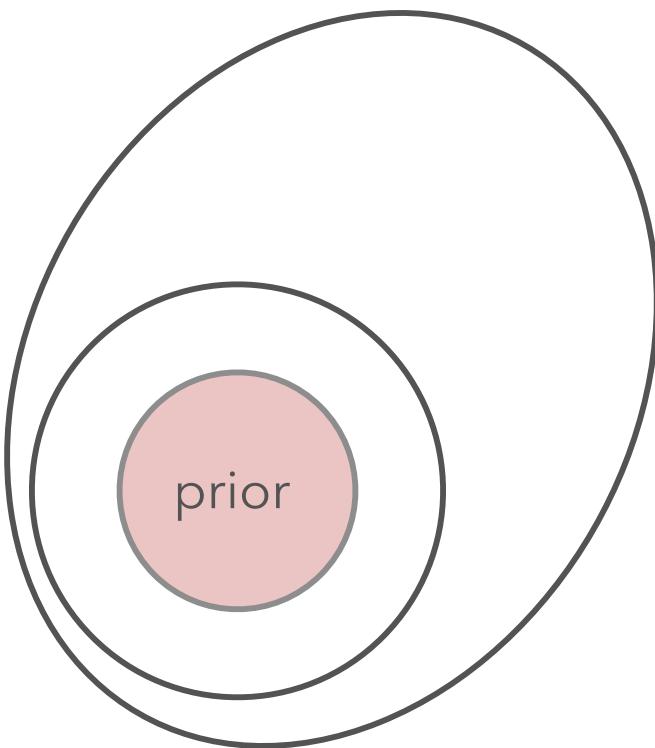
Early Stopping

- Uno de los métodos más simples y efectivos para prevenir overfitting en redes neuronales es *early stopping*: detener el entrenamiento después de pocas iteraciones o “apenas” se observa un empeoramiento del error de validación.



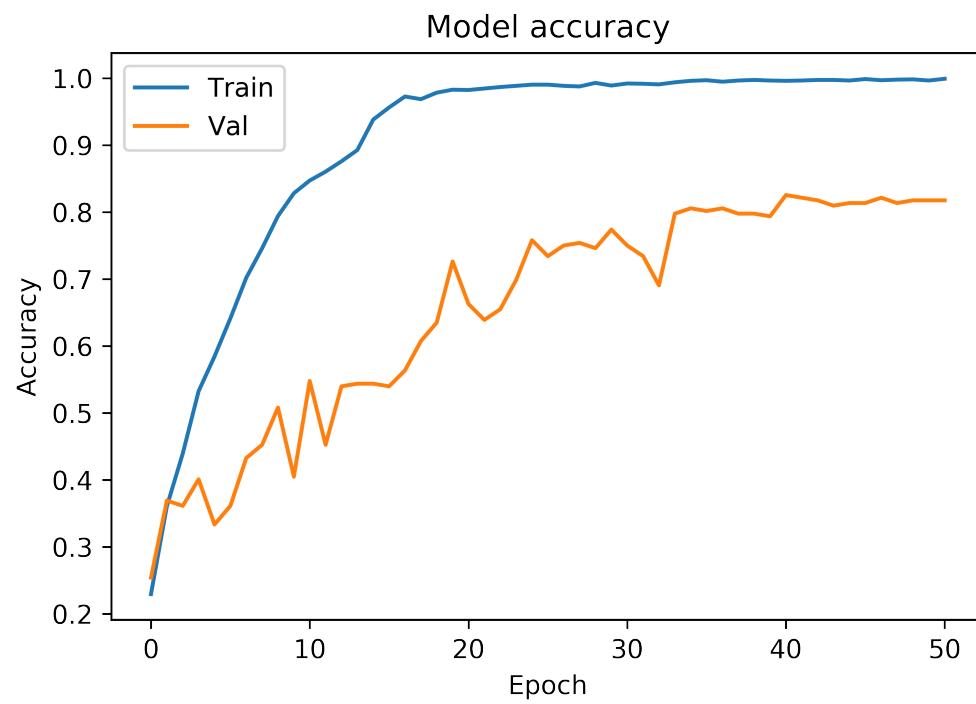
Early Stopping

- Intuición: Si la red es inicializada de manera consistente con algún prior, el entrenamiento la alejará de manera gradual de esa solución. A medida que el entrenamiento progresá la red (y sus parámetros) se harán cada vez más dependientes de los datos.



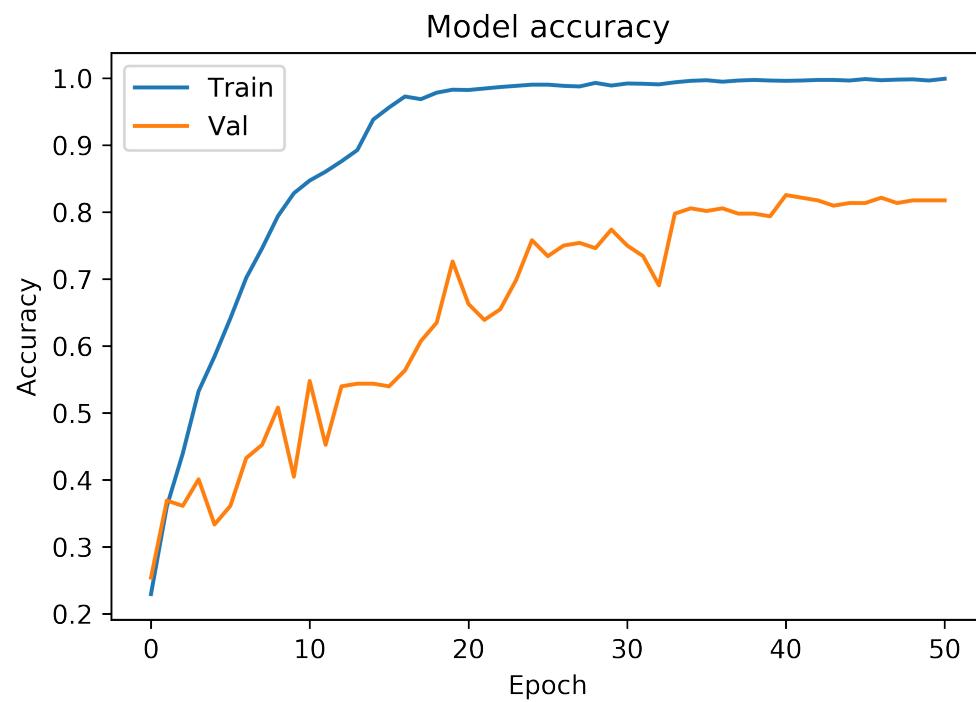
Early Stopping

- En la práctica esta técnica se suele operar con un conjunto de validación y con dos “mejoras” algorítmicas:
 1. Se considera una “pacienza” es decir un determinado número de iteraciones en que se permite que el error de validación empeore.



Early Stopping

- En la práctica esta técnica se suele operar con un conjunto de validación y con dos “mejoras” algorítmicas:
 2. Se conserva la mejor solución encontrada hasta un cierto momento (pocket algorithm).



Regularización vía Penalties

- Una gran familia de técnicas de regularización en machine learning se obtienen agregando *penalties* a la función objetivo optimizada por el modelo, es decir, términos que favorecen soluciones con ciertas características:

$$E(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(x^{(i)}; \theta), y^{(i)}) + \lambda \Omega(f(x; \theta))$$

- Ω se denomina regularizador y λ parámetro de regularización.
- Por ejemplo, los auto-encoders denominados “contractivos” que vimos en el capítulo anterior, optimizan el error de reconstrucción con un término adicional que penaliza fuertes variaciones en la activación de 1 o más capas escondidas ante pequeños cambios en la entrada

$$\Omega(f(x; \theta)) = \sum_K \left\| \frac{\partial h_i(x; \theta)}{\partial x} \right\|^2$$

Regularización L2 y Weight Decay

- Una elección muy común en redes neuronales (a veces denominada *Weight Decay*) consiste en penalizar la norma L2 de los pesos de conexión

$$E(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(x^{(i)}; \theta), y^{(i)}) + \sum_{\ell} \frac{\lambda_{\ell}}{2} \|W^{(\ell)}\|_2^2$$

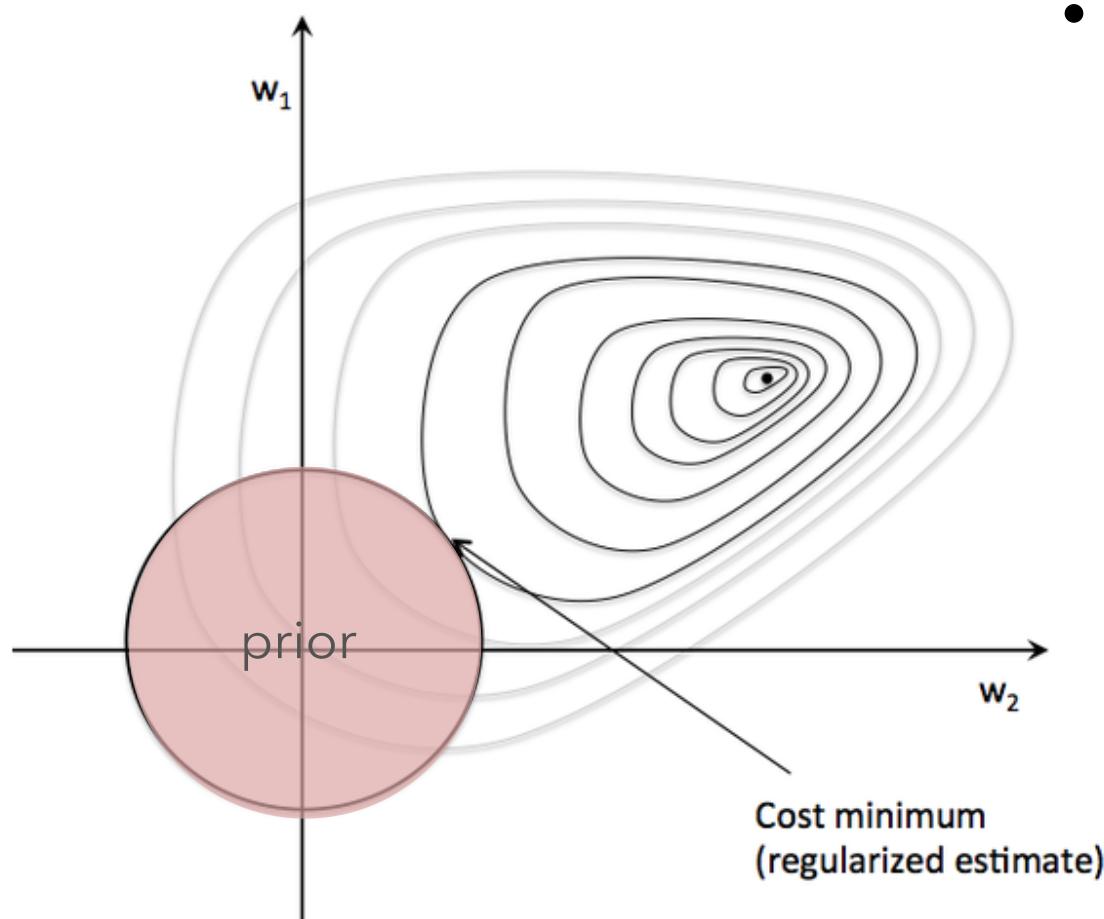
- En la interpretación Bayesiana, este regularizador es equivalente a elegir un prior Gaussiano para los pesos (con media nula y varianza independiente de los datos) y optimizar el a-posteriori.

$$P(\theta) \propto \exp(-\frac{\lambda}{2}\|\theta\|_2^2) \propto \mathcal{N}(0, \lambda^{-1}I)$$

$$\begin{aligned} \Rightarrow \log P(\theta|S) &\propto \log P(S|\theta) + \lambda \log P(\theta) \\ &= \ell(\theta) - \lambda\|\theta\|_2^2. \end{aligned}$$

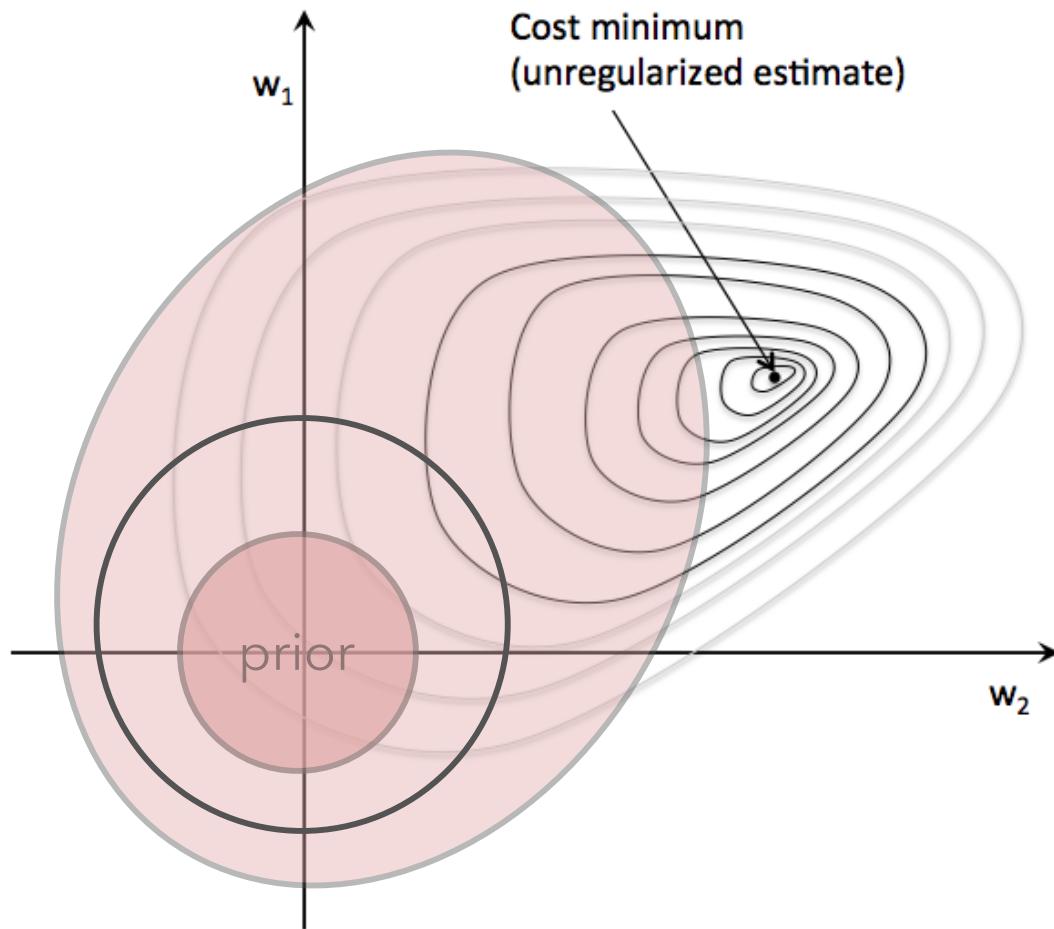


Regularización L2 y Weight Decay



- La elección $P(\theta) \propto \exp(-\frac{\lambda}{2}\|\theta\|_2^2) \propto \mathcal{N}(0, \lambda^{-1}I)$ busca codificar la preferencia de que, a no ser que sea necesario, esperamos que muchos de los parámetros del modelo sean 0, es decir, no sean *efectivamente utilizados* en el modelo.

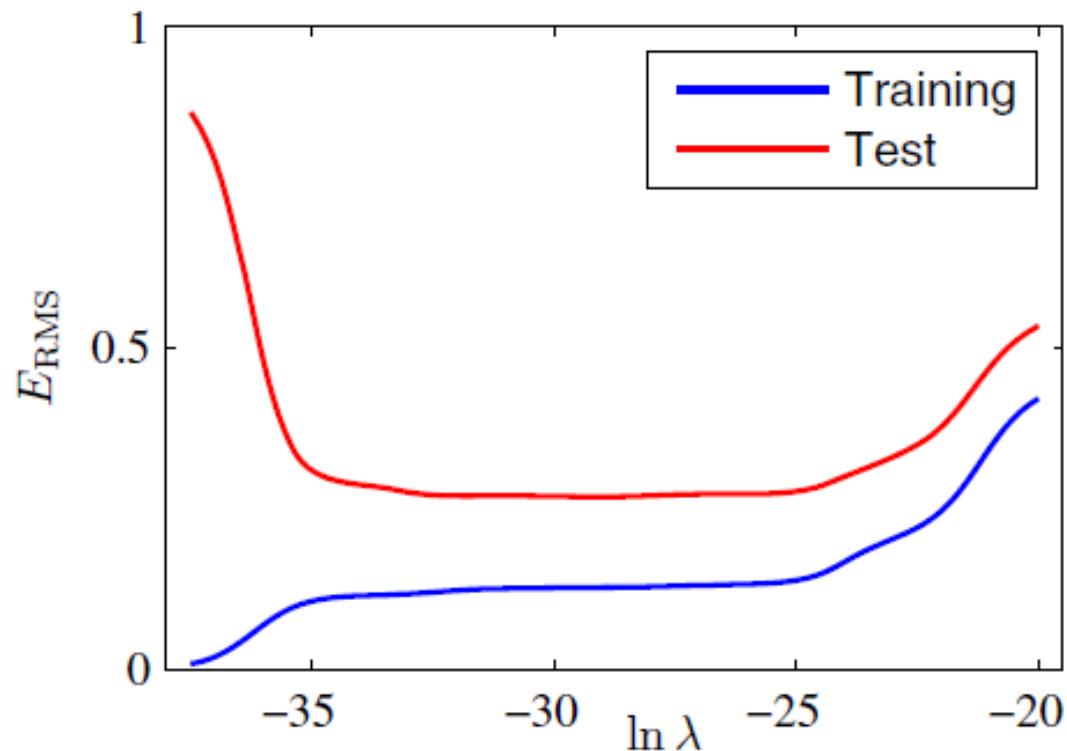
Regularización L2 y Weight Decay



- Disminuyendo el valor de λ disminuimos el grado de regularización impuesto al modelo, hasta que en algún momento no se restringe más a solución.
- El valor apropiado de λ depende del problema. ¿Podemos elegirlo para monitoreando el error de entrenamiento?

Regularización L2 y Weight Decay

- El valor apropiado de λ debe determinarse monitorizando un estimador del error de predicción. En el caso de la red es posible elegir valores diferentes por capa, pero hace necesario sintonizar múltiples parámetros.



Regularización L2 y Weight Decay

- Para entender mejor este método, usemos θ para denotar los parámetros de la red que se someten a *weight decay* sea θ^* el mínimo de la f.o. no regularizada $E(\theta)$ y consideremos una aproximación a segundo orden de $E(\theta)$ en torno a θ^*

$$\begin{aligned} E(\theta) &\approx E(\theta^*) + (\theta - \theta^*)^T \nabla E(\theta^*) + (\theta - \theta^*)^T H(\theta^*) (\theta - \theta^*) \\ &= E(\theta^*) + (\theta - \theta^*)^T H(\theta^*) (\theta - \theta^*) \end{aligned}$$

con $\nabla E_i = \partial E / \partial \theta_i$ y $H_{ij} = \partial^2 E / \partial \theta_i \partial \theta_j$.

- Con esta aproximación, la f.o. regularizada y su gradiente quedan

$$E_\lambda(\theta) \approx E(\theta^*) + (\theta - \theta^*)^T H(\theta^*) (\theta - \theta^*) + \lambda/2 \|\theta\|^2$$

$$\nabla E_\lambda(\theta) \approx H(\theta^*)(\theta - \theta^*) + \lambda \theta$$



Regularización L2 y Weight Decay

- Si ahora denotamos por θ_λ el mínimo de $E_\lambda(\theta)$, tenemos que

$$\nabla E_\lambda(\theta_\lambda) \approx H(\theta^*)(\theta_\lambda - \theta^*) + \lambda\theta_\lambda = 0$$

de modo que

$$\theta_\lambda \approx (H^* + \lambda I)^{-1} H^* \theta^*$$

si $H^* = H(\theta^*) = D = \text{diag}(\Delta_1^2, \Delta_2^2, \dots, \Delta_d^2)$ obtenemos

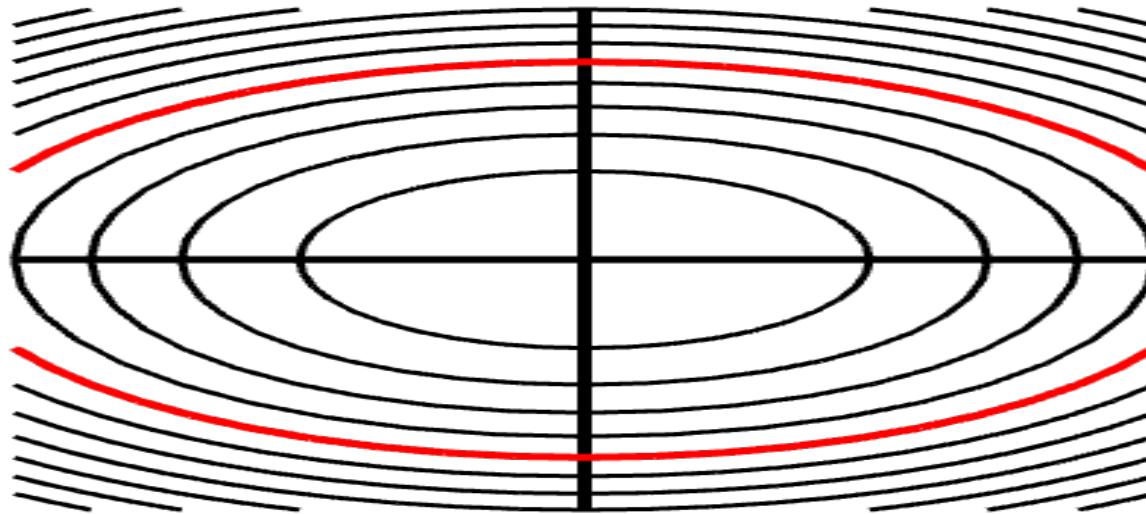
$$\theta_\lambda = (D + \lambda I)^{-1} D \theta^*$$

$$\Leftrightarrow \theta_{\lambda,i} = \left(\frac{\Delta_i^2}{\Delta_i^2 + \lambda} \right) \theta_i^* .$$



Regularización L2 y Weight Decay

$$\theta_{\lambda,i} = \left(\frac{\Delta_i^2}{\Delta_i^2 + \lambda} \right) \theta_i^*.$$



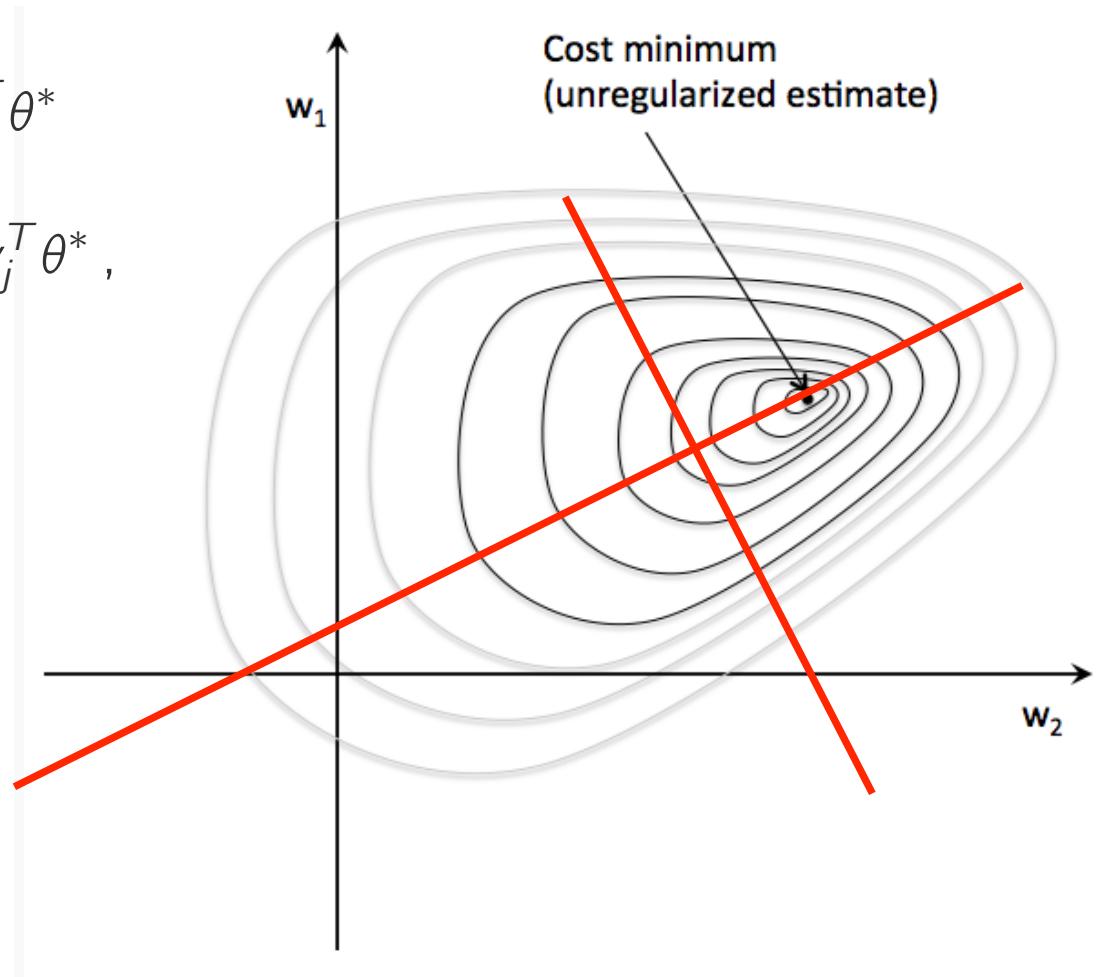
- Las direcciones más "podadas" por el regularizador son aquellas que corresponden a un valor pequeño de Δ^2 : direcciones del espacio de parámetros donde el gradiente del objetivo (no regularizado) no varía mucho y por lo tanto genera soluciones "parecidas" a la original (no regularizada).

Regularización L2 y Weight Decay

- Tomando ahora una EVD de H^* (no necesariamente diagonal)
 $H^* = VDV^T$, obtenemos

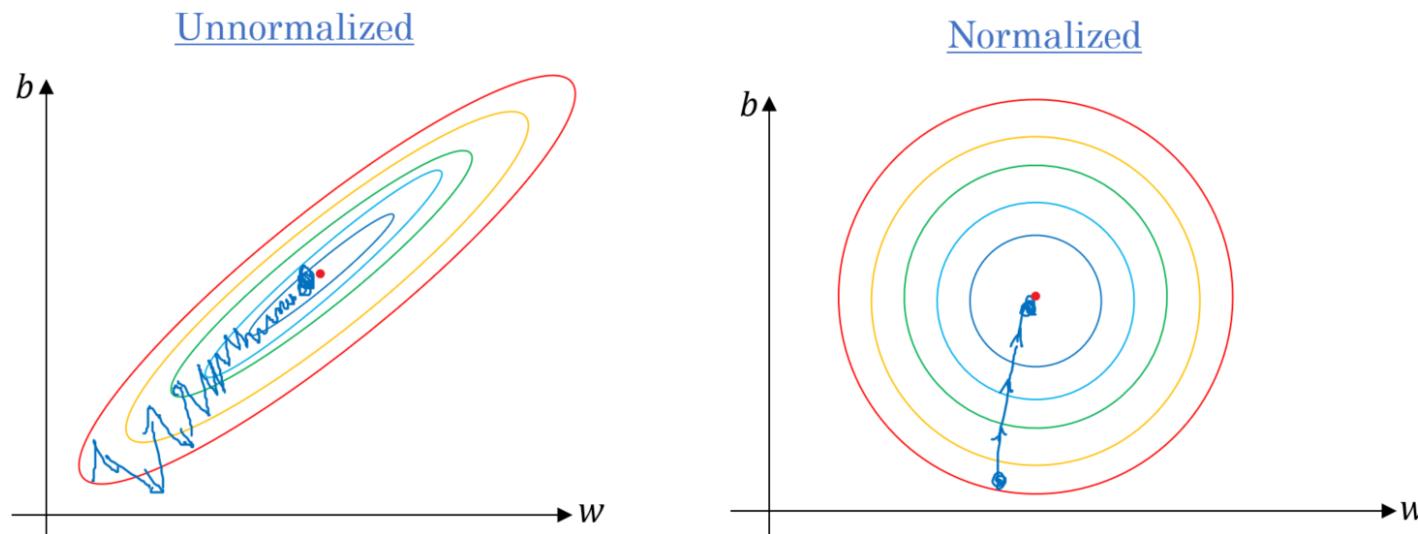
$$\theta_\lambda = V(D + \lambda I)^{-1} DV^T \theta^*$$

$$\theta_\lambda = \sum_j v_j \left(\frac{\Delta_j^2}{\Delta_j^2 + \lambda} \right) v_j^T \theta^*,$$



Regularización L2 y Weight Decay

- Las observaciones anteriores sugieren que esta forma de regularización podría ser efectiva no (o no sólo) restringiendo el espacio efectivo de modelos sino que “normalizando” la curvatura de la función objetivo. Esta situación ha sido sugerida/reportada por varios autores.



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.



Regularización L2 y Weight Decay

- El efecto de “poda” del método es bastante evidente cuando se considera la variación de la ecuación de actualización de los pesos usando este método. En efecto, si consideramos la f.o. regularizada para un parámetro

$$E_\lambda(\theta) = E(\theta) + \lambda/2\|\theta\|^2 \Rightarrow \nabla E_\lambda(\theta) = \nabla E(\theta) + \lambda\theta$$

obtenemos que el ajuste de la solución en el tiempo t , $\theta^{(t)}$, cambia como sigue:

$$\theta^{(t+1)} \leftarrow (1 - \lambda\eta_t)\theta^{(t)} - \eta_t \nabla E(\theta^{(t)})$$

donde λ es el parámetro de regularización o *decamiento*.

- Para algunos es ésta y sólo esta (en que se re-escala el valor de la solución antes del ajuste) la “implementación correcta” de weight decay.

Regularización L2 y Weight Decay

- Como muestra muy recientemente Zhang et al. en 2019, Weight Decay y la regularización con la norma L2 no resultan equivalentes para algunos de los métodos adaptivos de calibración de la tasa de aprendizaje que se utilizan regularmente en redes modernas. Estos métodos (e.g. Adam) aplican una regla de la forma

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta C^{-1} \nabla E(\theta^{(t)})$$

donde C es una cierta matriz.

Zhang, Guodong, et al. "Three mechanisms of weight decay regularization." ICLR 2019



Regularización L2 y Weight Decay

- En este caso, Weight Decay sería implementado mediante una regla de la forma:

$$\theta^{(t+1)} \leftarrow (1 - \lambda\eta)\theta^{(t)} - \eta C^{-1} \nabla E(\theta^{(t)})$$

mientras que la regularización L2 correspondería a

$$\theta^{(t+1)} \leftarrow (1 - \lambda\eta C^{-1})\theta^{(t)} - \eta C^{-1} \nabla E(\theta^{(t)})$$

los experimentos de los autores favorecen ampliamente a Weight Decay en una variedad de escenarios.



Zhang, Guodong, et al. "Three mechanisms of weight decay regularization."
ICLR 2019

Regularización L2 y Weight Decay

- Algunos de los resultados:

Dataset	Network	B	D	SGD		ADAM	
					WD		WD
CIFAR-10	VGG16	✓		83.20	84.87	83.16	84.12
			✓	86.99	88.85	88.45	88.72
CIFAR-10	ResNet32	✓		91.71	93.39	92.89	93.62
			✓	85.47	86.63	84.43	87.54
CIFAR-100	VGG16	✓	✓	86.13	90.65	89.46	90.61
	ResNet32	✓	✓	92.95	95.14	93.63	94.66
CIFAR-100	VGG16	✓	✓	68.42	73.31	69.88	74.22
CIFAR-100	ResNet32	✓	✓	73.61	77.73	73.60	77.40

Zhang, Guodong, et al. "Three mechanisms of weight decay regularization."
ICLR 2019



Entrenamiento con Ruido

- Otra técnica clásica de regularización de redes consiste en entrenar el modelo con “ruido” en los datos de entrenamiento.

Ejemplo original

$$(x, y)$$

Ejemplo “corrupto”

$$(x + \epsilon, y)$$

$$\mathbb{E}(\epsilon) = 0$$

$$\mathbb{E}(\epsilon\epsilon^T)_{ij} = \delta_{ij}\gamma^2$$

- Si consideramos un costo (loss) cuadrático, tenemos que la nueva función objetivo tiene la forma:

$$E_r = \mathbb{E}_\epsilon \left(\frac{1}{2n} \sum_i (f(x^{(i)} + \epsilon^{(i)}) - y^{(i)})^2 \right)$$

- Tomando nuevamente una aproximación en series de Taylor de $f(x + \epsilon)$...



Entrenamiento con Ruido

- Obtenemos

$$f(x + \epsilon) = f(x) + \sum_j \epsilon_j \frac{\partial f(x)}{\partial x_j} + \frac{1}{2} \sum_{j,k} \epsilon_j \epsilon_k \frac{\partial^2 f(x)}{\partial x_j \partial x_k} + \mathcal{O}(\epsilon^3)$$

- Notemos que (al tomar valor esperado en ϵ)

$$\mathbb{E}_\epsilon \left(\sum_j \epsilon_j \frac{\partial f(x)}{\partial x_j} \right) = 0$$

$$\mathbb{E}_\epsilon \left(\sum_{j,k} \epsilon_j \epsilon_k \frac{\partial^2 f(x)}{\partial x_j \partial x_k} \right) = \eta^2 \sum_j \frac{\partial^2 f(x)}{\partial x_j^2}$$

- De modo que al sustituir la aproximación en la f.o. E_r obtenemos ...

Entrenamiento con Ruido

$$E_r \approx \frac{1}{2n} \sum_i (f(x^{(i)} + \epsilon^{(i)}) - y^{(i)})^2 + \frac{\gamma^2}{2} \Omega(f)$$

con

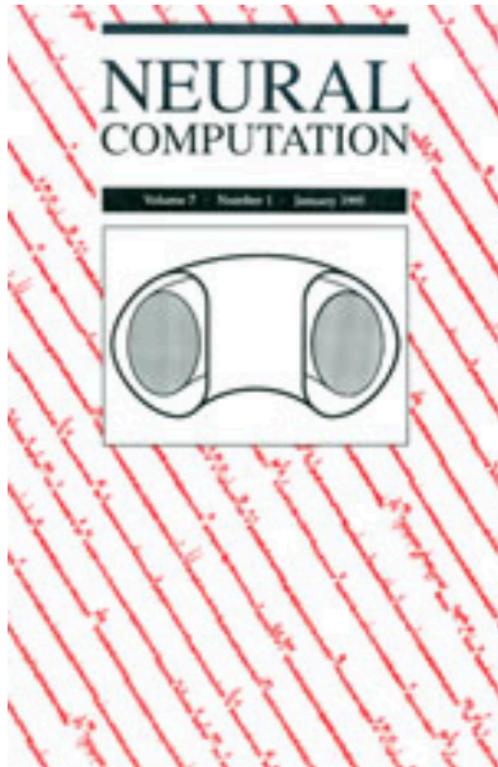
$$\begin{aligned}\Omega(f) &= \sum_i \sum_j \left(\frac{\partial f(x^{(i)})}{\partial x_j^{(i)}} \right)^2 + (f(x^{(i)}) - y^{(i)}) \sum_j \frac{\partial^2 f(x^{(i)})}{\partial x_j^{(i)2}} \\ &= \sum_i \left\| \nabla_{x^{(i)}} f(x^{(i)}) \right\|^2 + (f(x^{(i)}) - y^{(i)}) \text{Tr} \left(\nabla_{x^{(i)}}^2 f(x^{(i)}) \right)\end{aligned}$$

Si la red predice bien todos los ejemplos, el segundo término (verde), obtenemos solo

$$\Omega(f) = \sum_i \left\| \nabla_{x^{(i)}} f(x^{(i)}) \right\|^2$$



Entrenamiento con Ruido



Training with Noise is Equivalent to Tikhonov Regularization

1995

Chris M. Bishop

Posted Online April 04, 2008

<https://doi.org/10.1162/neco.1995.7.1.108>

© 1995 Massachusetts Institute of Technology

Neural Computation
Volume 7 | Issue 1 | January 1995
p.108-116

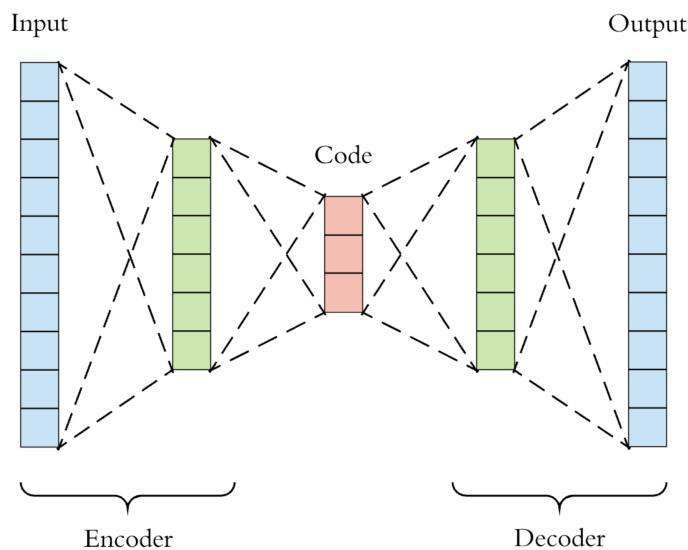


Entrenamiento con Ruido

- Notemos que este es esencialmente el mismo regularizador que usan los auto-encoder contractivos

$$\Omega(f(x; \theta)) = \sum_K \left\| \frac{\partial h_i(x; \theta)}{\partial x} \right\|^2$$

2011



Contractive Auto-Encoders: Explicit Invariance During Feature Extraction

Salah Rifai⁽¹⁾
Pascal Vincent⁽¹⁾
Xavier Muller⁽¹⁾
Xavier Glorot⁽¹⁾
Yoshua Bengio⁽¹⁾

⁽¹⁾ Dept. IRO, Université de Montréal, Montréal (QC), H3C 3J7, Canada

RIFAISAL@IRO.UMONTREAL.CA
VINCENTP@IRO.UMONTREAL.CA
MULLERX@IRO.UMONTREAL.CA
GLOROTXA@IRO.UMONTREAL.CA
BENGIOY@IRO.UMONTREAL.CA

Abstract

We present in this paper a novel approach for training deterministic auto-encoders. We show that by adding a well chosen penalty term to the classical reconstruction cost function, we can achieve results that equal or surpass those attained by other regularized auto-encoders as well as denoising auto-encoders on a range of datasets. This penalty term

1. Introduction

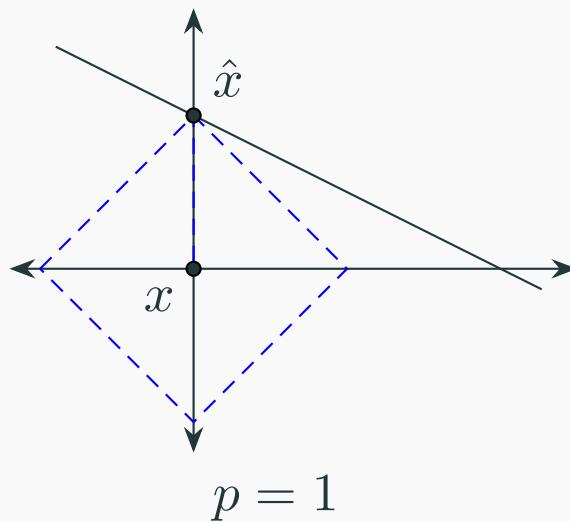
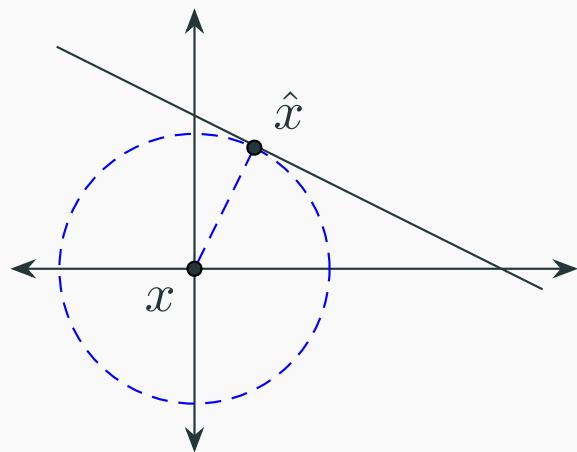
A recent topic of interest¹ in the machine learning community is the development of algorithms for unsupervised learning of a useful representation. This automatic discovery and extraction of features is often used in building a deep hierarchy of features, within the contexts of supervised, semi-supervised, or unsupervised modeling. See Bengio (2009) for a recent review of Deep Learning algorithms. Most of these



Regularización L1

- Una alternativa clásica a la norma L2 es la norma L1

ℓ_p norms: $\|\theta\|_{\ell_p} = \sqrt[p]{\sum_i \theta_i^p}$ (cuando $p \in [0, 1]$, pseudo-normas).

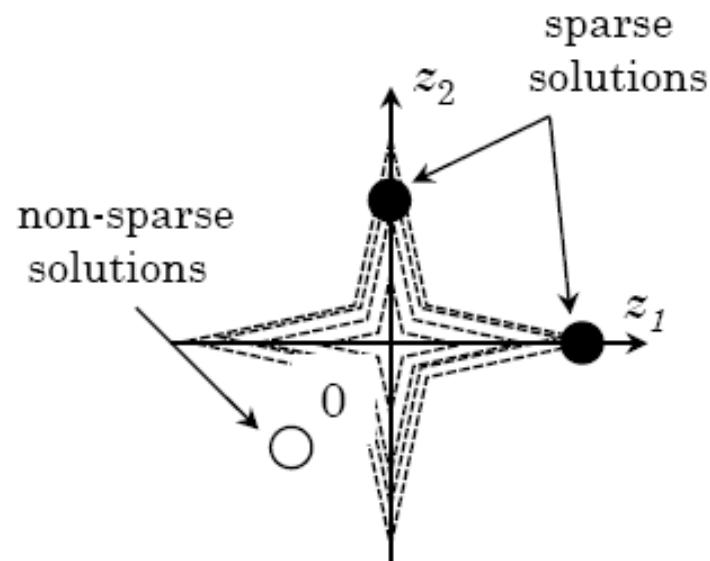


Regularización L1

- De entre las normas ℓ_p , la norma ℓ_1 es la más "cercana" a la norma ℓ_0 que cumple con ser convexa (gran ventaja desde el punto de vista de la optimización).

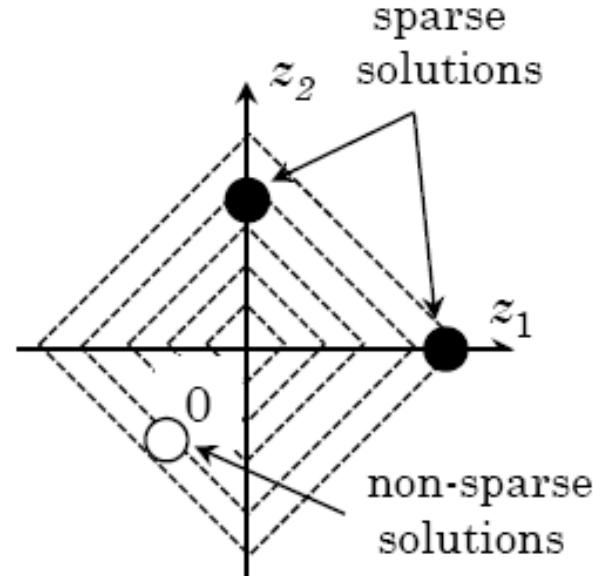
$$\|\theta\|_{\ell_0} = \sum_i I(\theta_i \neq 0)$$

L_0 norm



$$\|\theta\|_{\ell_1} = \sum_i |\theta_i|$$

L_1 norm



Regularización L1

- La función objetivo de la red toma la forma

$$E(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(x^{(i)}; \theta), y^{(i)}) + \sum_{\ell} \lambda_{\ell} \|W^{(\ell)}\|_1$$

$$\|W^{(\ell)}\|_1 = \sum_{ij} |W_{ij}^{(\ell)}|$$

- Lamentablemente este penalty no es estrictamente diferenciable. Sin embargo es posible sustituir el *gradiente* de $\Omega(\theta) = |\theta|$ (necesario para el entrenamiento de la red) por su *sub-gradiente*, i.e. una dirección g tal que

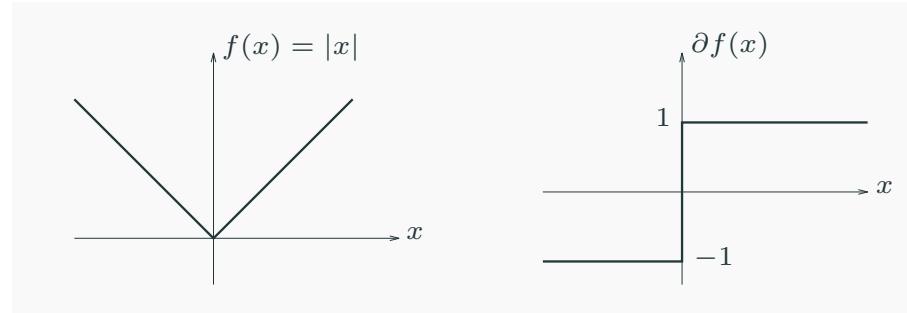
$$\Omega(\theta) \geq \Omega(\theta_0) + g^T (\theta - \theta_0)$$

es decir, una dirección que permite replicar al gradiente (o su negativo) como dirección de ascenso (descenso).

Regularización L1

- Para la norma ℓ_1 es dirección g viene dada por el vector de componentes

$$\partial|w_i| = \begin{cases} +1 & w_i > 0 \\ -1 & w_i < 0 \\ [-1, 1] & w_i = 0 \end{cases}$$

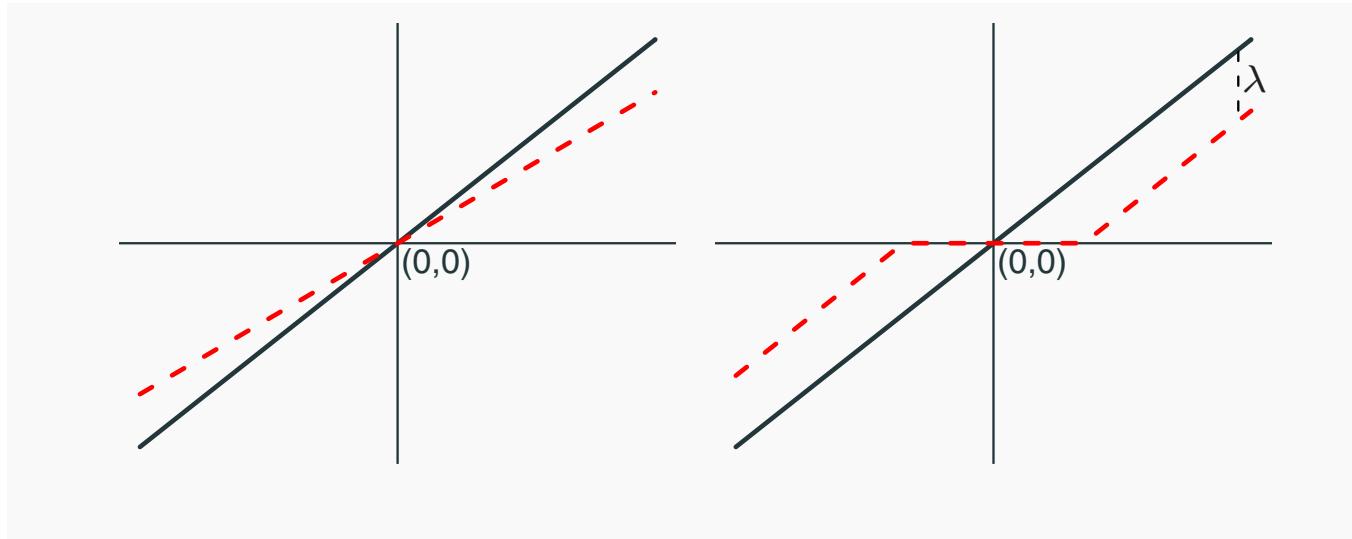


- No es difícil ver que si la norma L2 tiene un efecto de poda multiplicativo, la norma L1 tiene una efecto de poda aditivo. Para una Hessiana diagonal obtenemos

$$\theta_{\lambda,i} = \text{sign}(\theta_i^*) \max \left(\theta_i^* - \frac{\lambda}{\Delta_i^2}, 0 \right).$$

Regularización L1

$$\theta_{\lambda,i} = \text{sign}(\theta_i^*) \max \left(\theta_i^* - \frac{\lambda}{\Delta_i^2}, 0 \right).$$



- En la práctica esto lo hace mucho más efectivo para llevar los pesos a 0, pero altera de modo más violento las direcciones en que el algoritmo avanzaría sin el regularizador, retardando el aprendizaje (L2 sólo escala).

Regularización con la Norma Espectral

- Recientemente también Yoshida & Miyato (2017) muestran que penalizar la norma espectral de los pesos resulta mucho más efectivo y apropiado para propósitos de regularización que la norma L2 (Frobenius).

$$\|W^{(\ell)}\|_\rho = \max_a \frac{\|W^{(\ell)} a\|_2}{\|a\|_2}$$

- Para ξ suficientemente pequeño:

$$\frac{\|f(x + \xi) - f(x)\|_2}{\|\xi\|_2} \leq \|W^{(\ell)}\|_\rho$$

- Es posible calcular la norma espectral encontrando el valor propio más grande la matriz de pesos, tarea para la que existen algoritmos relativamente eficientes.

Yoshida, Yuichi, and Takeru Miyato. "Spectral norm regularization for improving the generalizability of deep learning." *arXiv preprint arXiv:1705.10941* (2017).



Regularización con la Norma Espectral

- Algunos de los resultados:

Table 1: Test accuracy and generalization gap (best values in bold).

Model	B	Test accuracy				α	Generalization gap			
		vanilla	decay	adver.	spectral		vanilla	decay	adver.	spectral
VGGNet	64	0.898	0.897	0.884	0.904	0.88	0.079	0.074	0.109	0.068
	4096	0.858	0.863	0.870	0.885	0.85	0.092	0.064	0.064	0.045
NIN	64	0.626	0.672	0.627	0.669	0.62	0.231	0.120	0.253	0.090
	4096	0.597	0.618	0.607	0.640	0.59	0.205	0.119	0.196	0.090
DenseNet (CIFAR100)	64	0.675	0.718	0.675	0.709	0.67	0.317	0.080	0.299	0.095
	4096	0.639	0.671	0.649	0.697	0.63	0.235	0.111	0.110	0.051
DenseNet (STL-10)	64	0.724	0.723	0.707	0.735	0.70	0.063	0.073	0.069	0.068
	4096	0.686	0.689	0.676	0.697	0.67	0.096	0.057	0.015	0.042



Yoshida, Yuichi, and Takeru Miyato. "Spectral norm regularization for improving the generalizability of deep learning." *arXiv preprint arXiv:1705.10941* (2017).

Entonces ...

- Denominaremos overfitting a la situación indeseable en que el error de predicción del modelo es mucho más grande que su error en los casos de entrenamiento. Diagnosticar y provenir este problema es importante.
- La práctica estándar consiste en reservar un conjunto de pruebas en que evaluar el error de predicción del modelo y un conjunto de validación para seleccionar el modelo correcto i.e. monitorear el error de predicción obtenido mediante diferentes modificaciones de la arquitectura.
- Los métodos para prevenir overfitting antes de evaluar en el conjunto de pruebas se denominan **métodos de regularización**.
- Algunas técnicas clásicas para afrontar este problema en redes neuronales son: early stopping, regularización L2, weight decay, entrenamiento con ruido y regularización L1. Todos ellos se pueden interpretar como la introducción de un prior sobre el espacio de soluciones que codifica la preferencia de base por modelos simples, con pocos parámetros activos o “estables” en términos de variaciones en los datos de entrenamiento.

