

Tópicos Avanzados en Entrenamiento de Redes Profundas

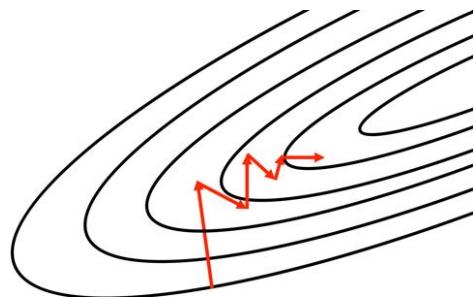
Inicialización y Batch-Normalization



Prof. Ricardo Ñanculef - Departamento de Informática UTSMS

Inicialización

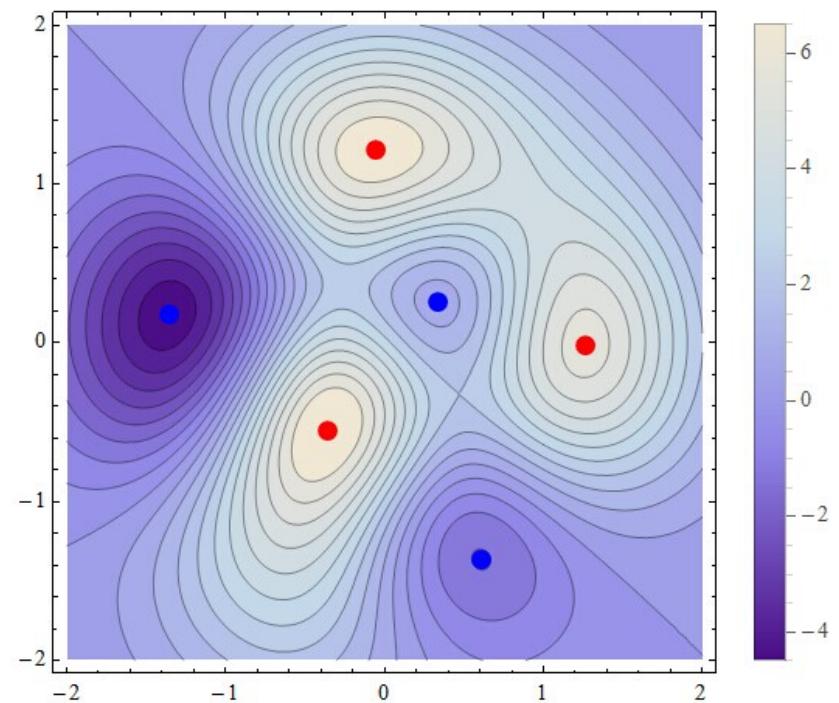
- Dado que el algoritmo que usamos para entrenar la red es iterativo, necesitamos un criterio para dar valores iniciales a los parámetros. Este paso puede influir fuertemente en el resultado obtenido.



- Problema: $\min_w E(\mathbf{w})$
- Algoritmo:
1 **for** $t = 1, \dots, T$ **do**
2 | $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} - \eta \frac{\delta E}{\delta \mathbf{w}}$;
3 **end**

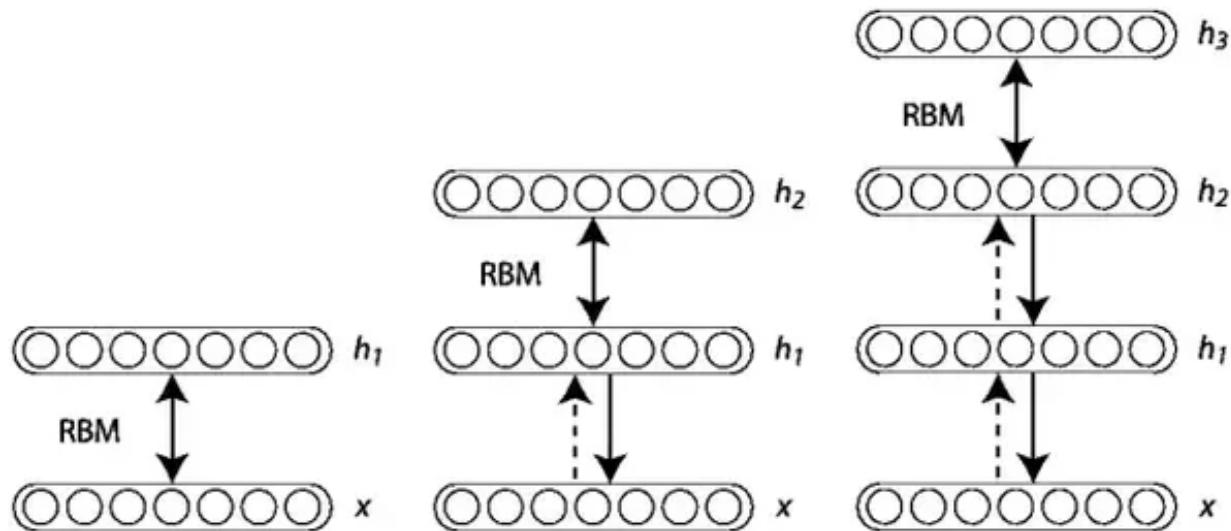
Inicialización

- Esto es particularmente cierto en redes que son poco profundas ya que en este caso los mínimos locales de la f.o. pueden diferir mucho en términos de error de predicción.



Inicialización

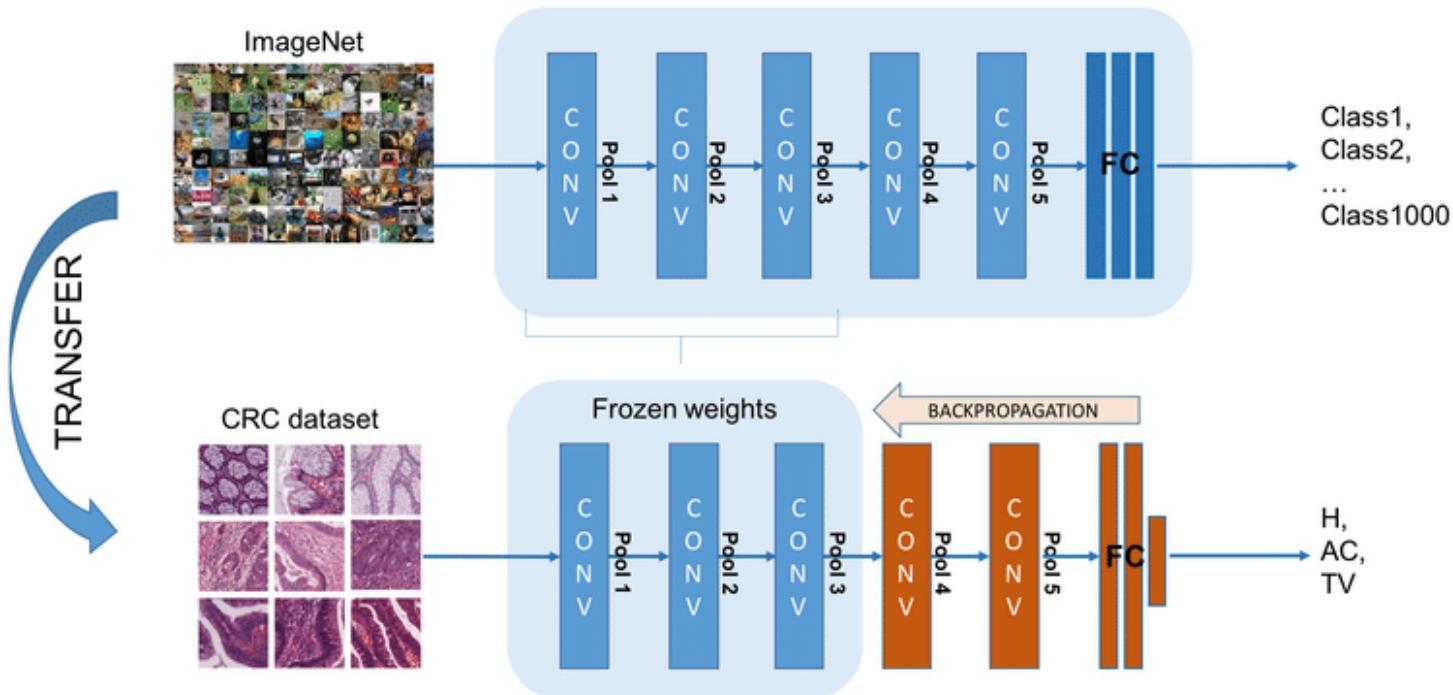
- En redes profundas, esto último no es un problema, pero, como hemos discutido, el flujo de las señales en el grafo puede ser inestable (explosión, desvanecimiento). Una inicialización adecuada puede ayudar a atenuar este problema.



Hinton, Geoffrey E. et al. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.

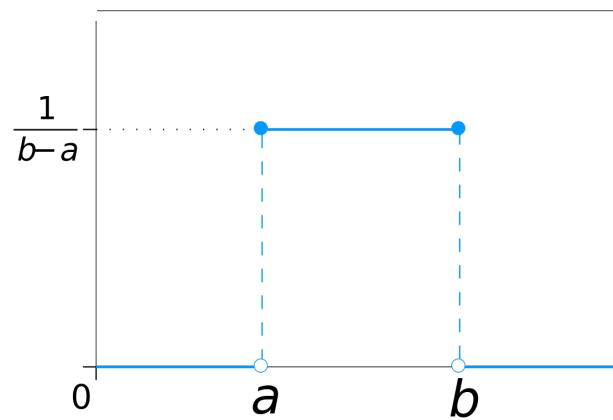
Inicialización

- La idea de utilizar redes pre-entrenadas en grandes datasets puede en efecto verse como un método de inicialización.

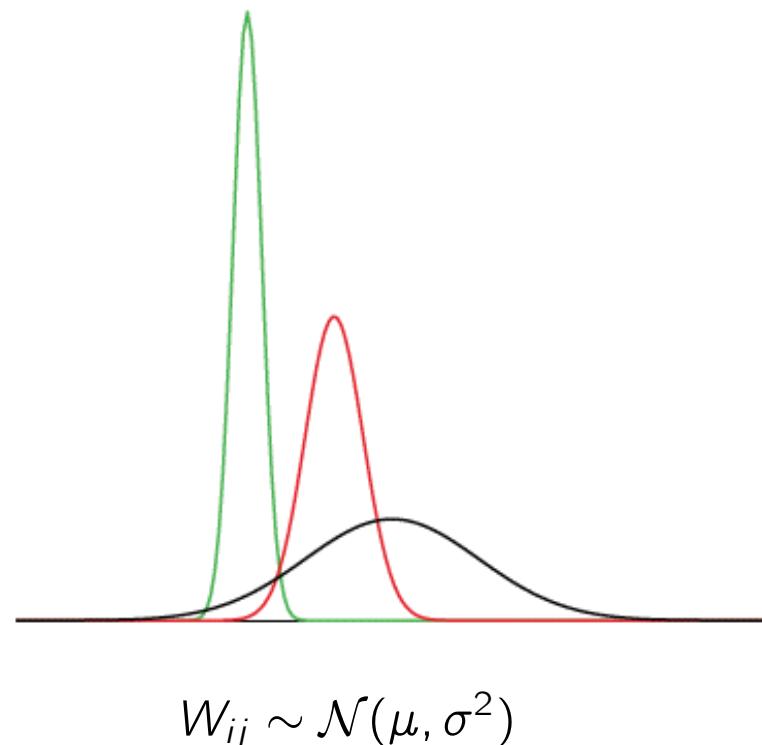


Inicialización Aleatoria

- La estrategia clásica para inicializar los pesos de la red consiste en muestrear una distribución de alta entropía como una uniforme o una Gaussiana.



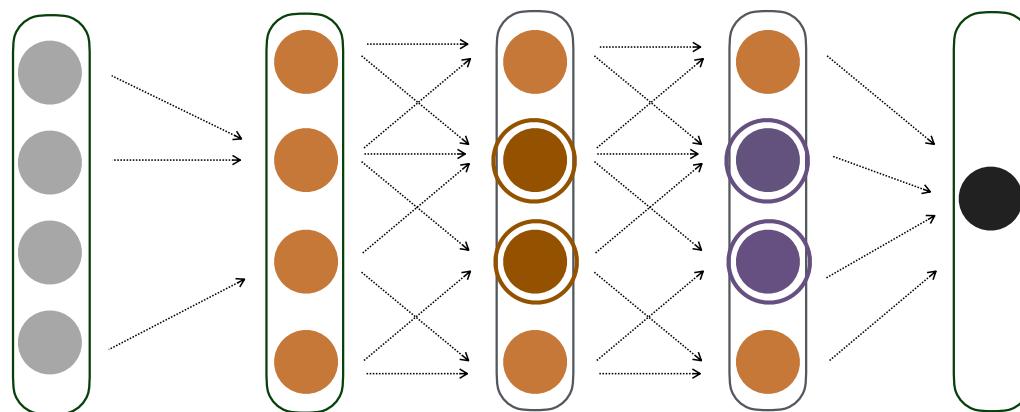
$$W_{ij} \sim U(a, b)$$



$$W_{ij} \sim \mathcal{N}(\mu, \sigma^2)$$

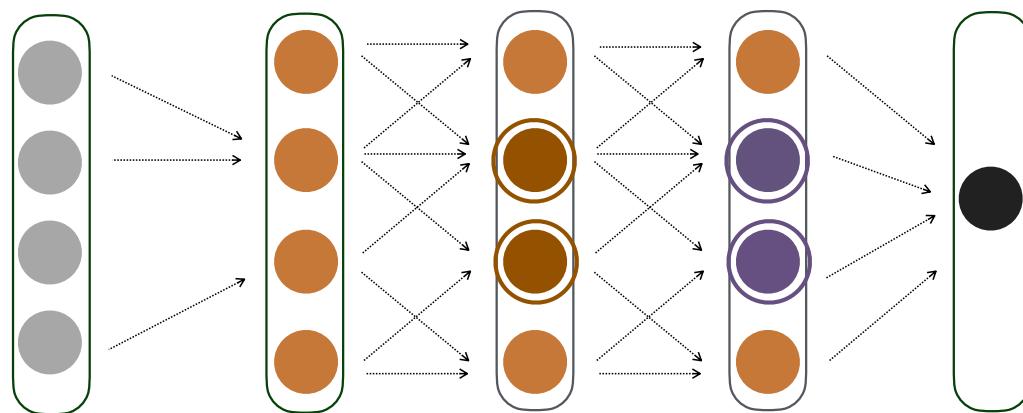
Symmetry Breaking

- Esto ayuda a evitar el problema de la co-adaptación, es decir la especialización de dos o más unidades de la red en la misma tarea, situación que ocurre con facilidad en capas con mucha simetría en sus patrones de conectividad.



Symmetry Breaking

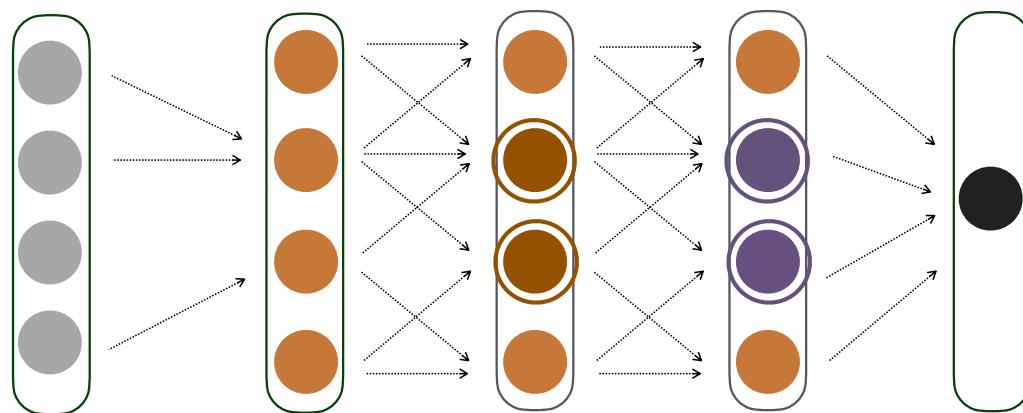
- Esto ayuda a evitar el problema de la co-adaptación, es decir la especialización de dos o más unidades de la red en la misma tarea, situación que ocurre con facilidad en capas con mucha simetría en sus patrones de conectividad.



Si las dos unidades en cafés se inicializan con los mismos pesos, transmitirán exactamente el mismo atributo a la capa siguiente para cada entrada. Esta redundancia no es necesaria si las unidades de la capa siguiente pueden "leer" el atributo desde cualquiera de las unidades.

Symmetry Breaking

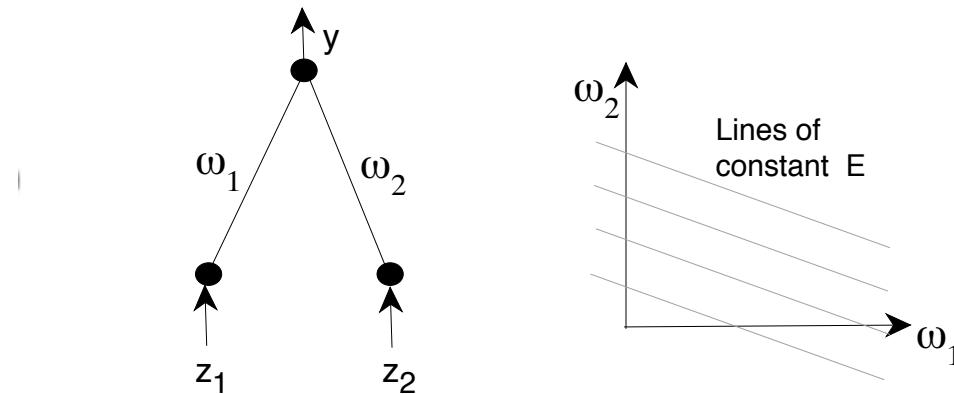
- Esto ayuda a evitar el problema de la co-adaptación, es decir la especialización de dos o más unidades de la red en la misma tarea, situación que ocurre con facilidad en capas con mucha simetría en sus patrones de conectividad.



Si, además, las unidades en verde se inicializan con los mismos pesos, las unidades amarillas recibirán los mismos ajustes durante el entrenamiento, perpetuando la redundancia.

Symmetry Breaking

- Esta co-adaptación no sólo representa una pérdida de eficiencia representacional, sino que puede hacer el espacio de búsqueda más difícil de navegar usando algoritmos de primer orden (entrenamiento más difícil).



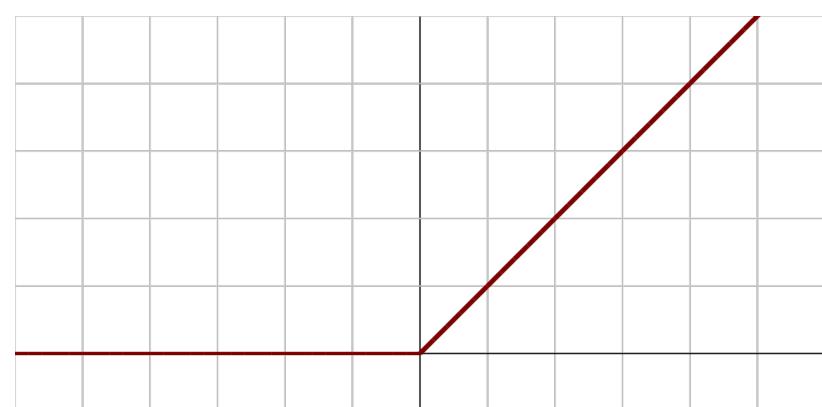
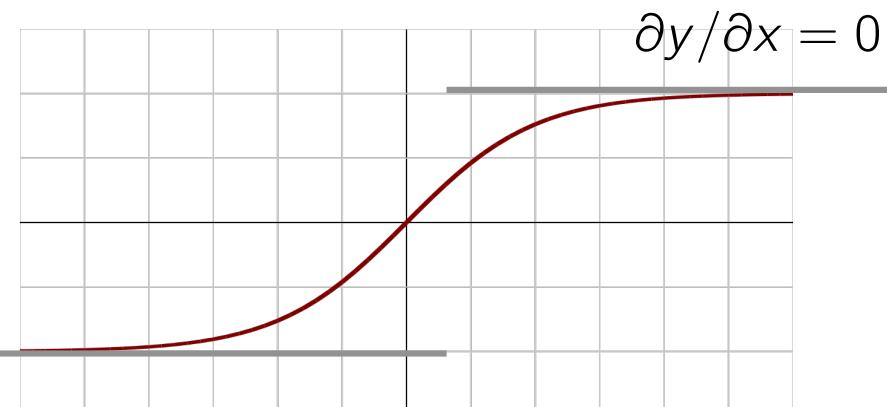
$$w_1 z_1 + w_2 z_2 = w_1 z_1 + w_2 z_1 = (w_1 + w_2) z_1$$

$$(w_1 + w_2) = \text{cte}$$

La existencia de atributos perfectamente correlacionados linealmente, situación que crea sub-espacios completos del espacio de parámetros con el mismo valor de la función objetivo.

Inicialización Aleatoria - Media

- Una vez elegida la familia, los parámetros de la distribución usualmente se determinan para obtener una determina media y varianza.
- Como la inmensa mayoría de las funciones de activación tienen su rango útil de sensibilidad en torno a 0, elegir una media 0 ayuda a que las neuronas no partan saturadas (por lo tanto puedan aprender más rápidamente)

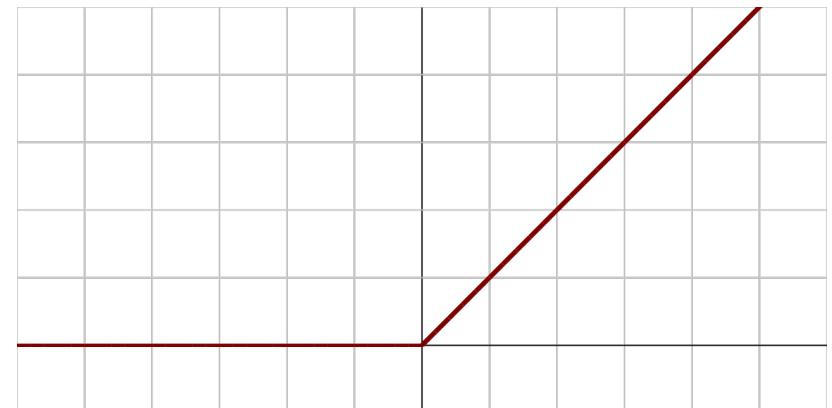
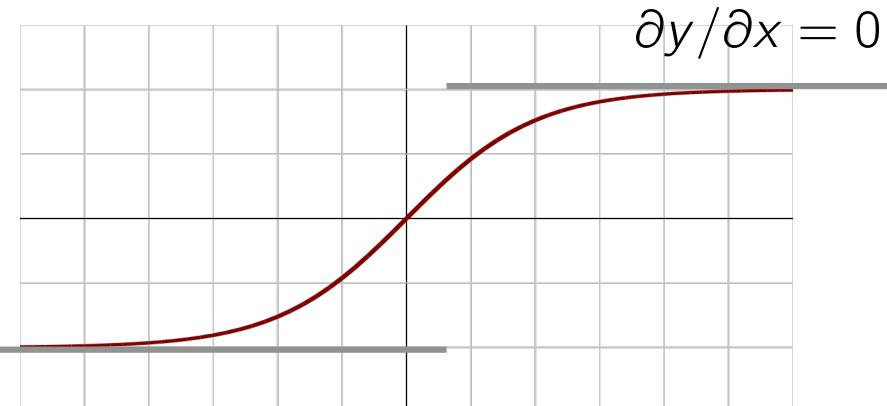


$$y = g(w^T x - b)$$

$$a^{(\ell)} = g(W^{(\ell)} a^{(\ell-1)} - b^{(\ell)})$$

Inicialización Aleatoria - Media

- Esta elección se suele sincronizar con una normalización de los datos que nos ayude recibir pre-activaciones centradas en 0.

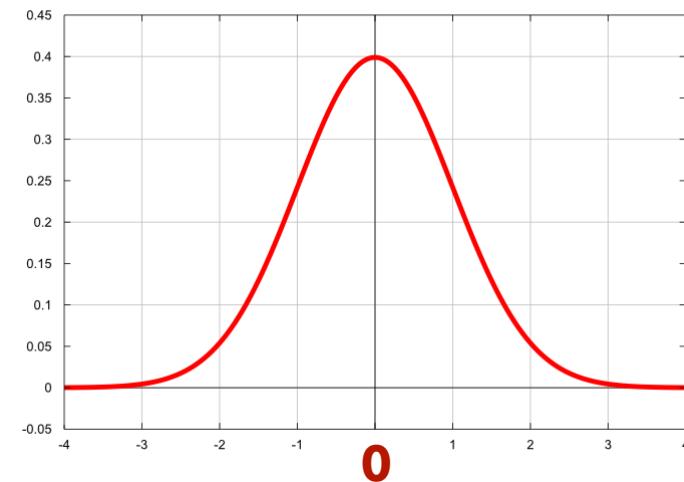
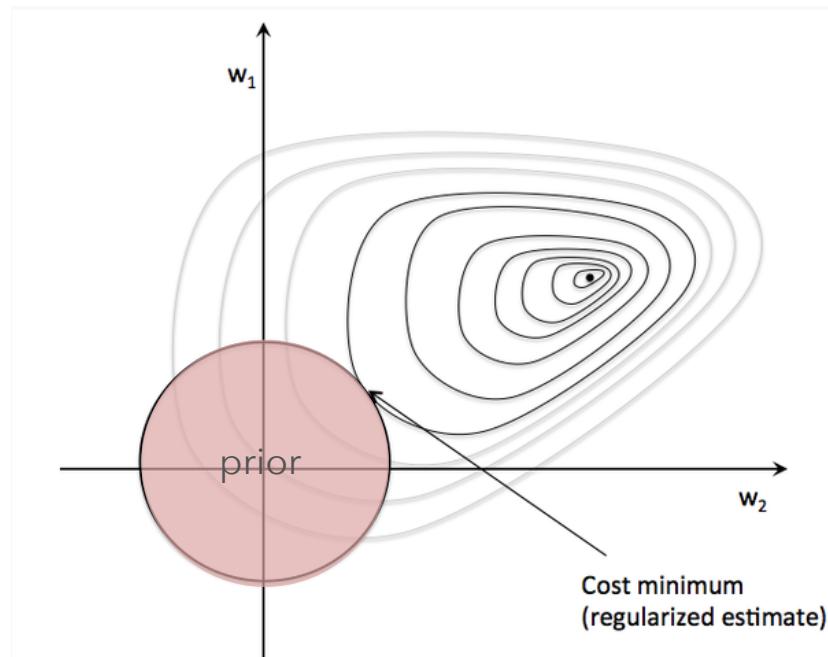


$$y = g(w^T x - b)$$

$$\mathbb{E}(p) = \mathbb{E}(w^T x - b) \approx \mathbb{E}(w)^T \mathbb{E}(x)$$

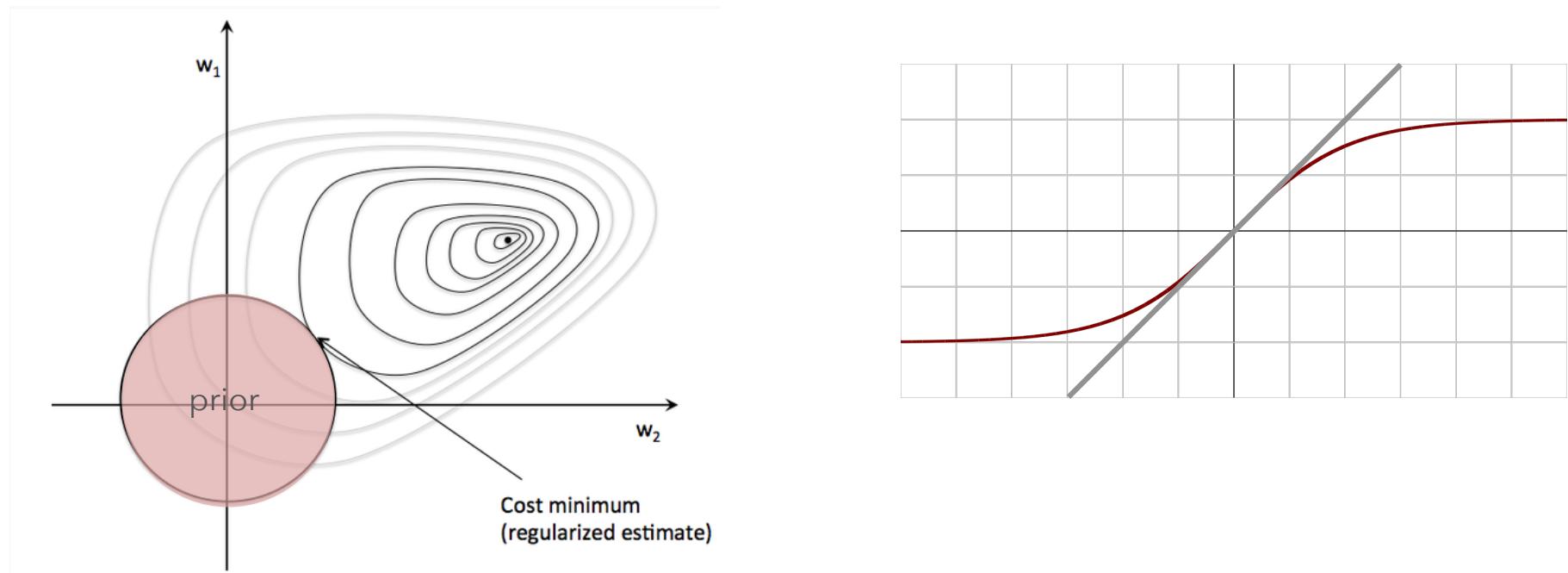
Inicialización Aleatoria - Media

- La media nula es también consistente con el *prior* más común en regularización: un peso debiese ser cero a no ser que sea necesario.



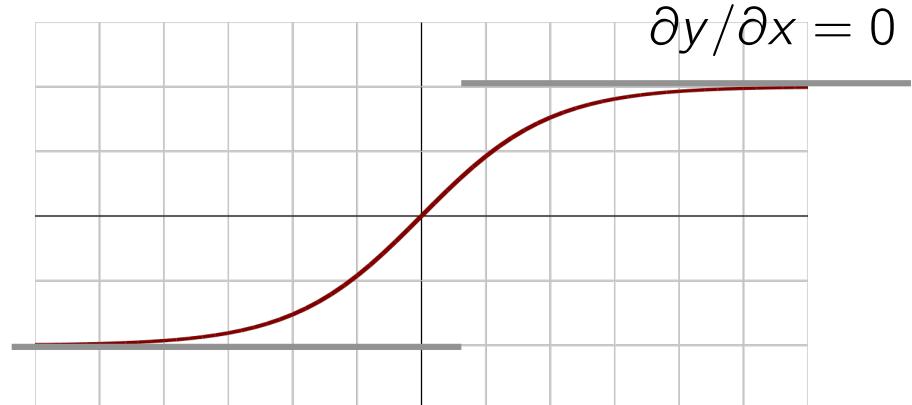
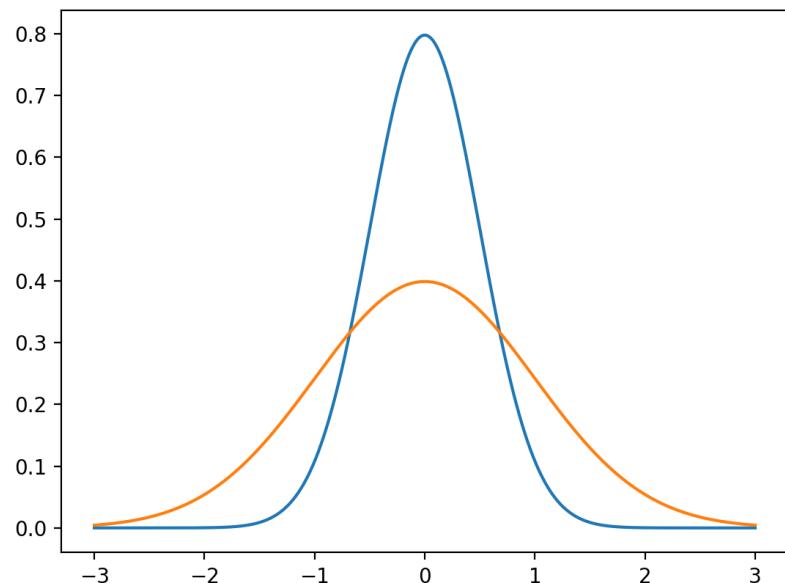
Inicialización Aleatoria - Media

- En redes con funciones de activación clásicas (S-shaped), esa elección coincide felizmente con la preferencia por modelos lineales.



Inicialización Aleatoria - Varianza

- La varianza de la distribución debiese ser grande para romper la simetría de las unidades, pero tampoco demasiado grande ya que esto podría ocasionar saturación prematura de unidades (dead units), tanto de la unidad que recibe esos pesos como de las unidades a quienes ésta última está conectada.

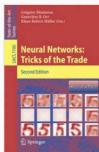


Inicialización Aleatoria - Varianza

- LeCun & Co proponen muestrear una uniforme con los siguientes parámetros

$$W_{ij} \sim U\left(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right)$$

donde m es la dimensionalidad de entrada de la neurona involucrada.

 Neural Networks: *Tricks of the Trade* pp 9-48 | [Cite as](#)

Efficient BackProp

Authors [Yann A. LeCun](#), [Léon Bottou](#), [Genevieve B. Orr](#), [Klaus-Robert Müller](#)

Chapter   
Citations Mentions Downloads

Part of the [Lecture Notes in Computer Science](#) book series (LNCS, volume 7700)

Abstract

The convergence of back-propagation learning is analyzed so as to explain common phenomenon observed by practitioners. Many undesirable behaviors of backprop can be avoided with tricks that are rarely exposed in serious technical publications. This paper gives some of those tricks, and offers explanations of why they work.



LeCun, Yann A., et al. "Efficient backprop." *Neural Networks: Tricks of the Trade*. Springer, Berlin, Heidelberg, 1998 9-48 (Re-editado y re-publicado en 2012)

Inicialización Aleatoria - Varianza

- Notemos que si $w \sim \mathcal{U}(a, b)$, su varianza está dada por

$$\text{Var}(w) = \frac{(b - a)^2}{12}$$

$$-a = b = 1/m \Rightarrow \text{Var}(w) = \frac{1}{3m} \propto \frac{1}{m}$$



Inicialización Aleatoria - Varianza

- Notemos que si $w \sim \mathcal{U}(a, b)$, su varianza está dada por

$$\text{Var}(w) = \frac{(b - a)^2}{12}$$

$$-a = b = 1/m \Rightarrow \text{Var}(w) = \frac{1}{3m} \propto \frac{1}{m}$$

- Si consideramos una neurona con ecuación $y = g(w^T x - b)$, y asumimos que los pesos y los datos de entrada son aproximadamente independientes, tenemos que

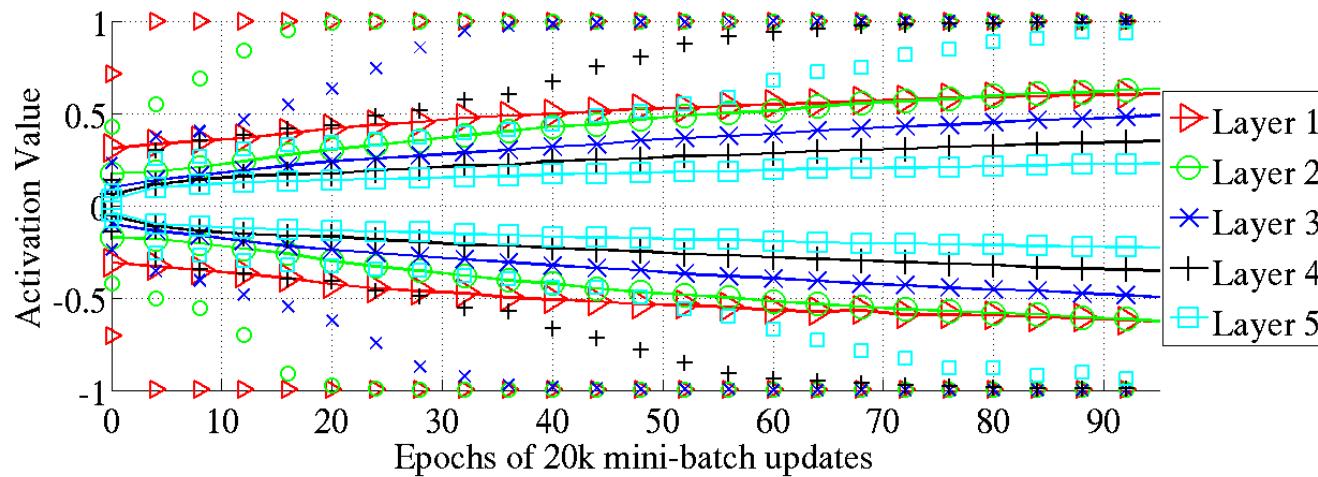
$$\sigma_y^2 = \text{var}(y) \approx \sum_{i=1}^m \text{var}(x_i) \text{var}(w_i) \approx m\sigma_x^2\sigma_w^2$$

donde σ_x^2 es la varianza de los atributos de entrada.

En otras palabras, con la elección de LeCun, la neurona preserva la varianza de los atributos de entrada.

Inicialización Aleatoria - Varianza

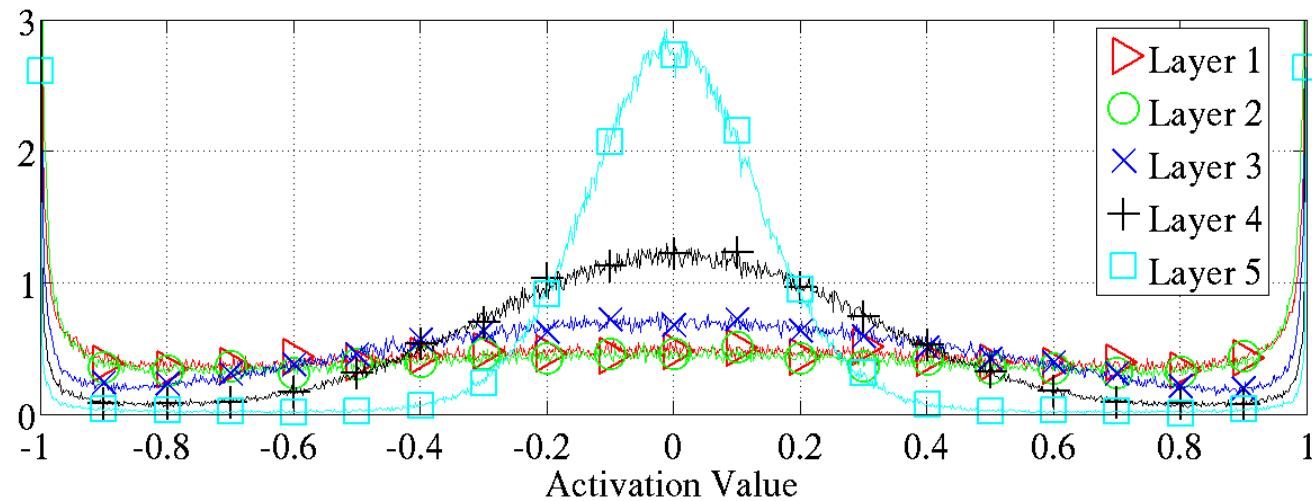
- Como observa Glorot en 2010, esta estrategia no impide que las varianzas cambien notablemente de capa en capa a medida que el entrenamiento progresá. De hecho sucede algo “insperado”: las varianzas son mucho más altas para las capas más bajas en la red (abajo: evolución de los perceptibles 2 y 98 correspondientes a las activaciones de diferentes capas de una red con neuronas tanh)



Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Inicialización Aleatoria - Varianza

- El histograma siguiente muestra las activaciones al final del entrenamiento en una red de 5 capas con neuronas tanh inicializadas con el método clásico.



Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Inicialización Aleatoria - Varianza

- Glorot propone en cambio la siguiente regla:

$$W_{ij} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right)$$

donde n es el número de unidades de la capa de la que la neurona forma parte.



Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Inicialización Aleatoria - Varianza

- Idea: para que se mantenga la varianza no sólo es importante considerar el forward pass, sino que también en el backward pass.

$$\delta_i^{(\ell)} = \sum_t \delta_t^{(\ell+1)} g'(p^{(\ell+1)})_t W_{ti}^{(\ell+1)}$$

||

$$\frac{\partial L(f(x), y)}{\partial W_{ij}^{(\ell)}} = \boxed{\frac{\partial L(f(x), y)}{\partial a_i^{(\ell)}}} \boxed{\frac{\partial a_i^{(\ell)}}{\partial W_{ij}^{(\ell)}}}$$

||

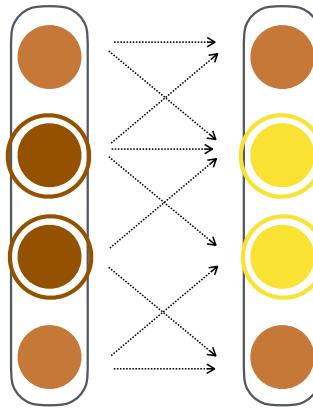
$$g'(p_i^{(\ell)}) a_j^{(\ell-1)}$$

$$p^{(\ell)} = W^{(\ell)} a^{(\ell-1)} - b^{(\ell)}$$

Inicialización Aleatoria - Varianza

- Simplificando un poco la notación:

$$\delta_i^{(\ell-1)} = \sum_j \delta_j^{(\ell)} g'_j(\cdot) w_{ji}$$



$$y_j = g(w^T x - b)$$

$$\sigma_{\delta^{(\ell-1)}}^2 = \text{var}(\delta_i^{(\ell-1)}) \approx \sum_j \text{var}(\delta_j^{(\ell)}) \text{var}(w_{ji}) = n^{(\ell)} \sigma_{\delta^{(\ell)}}^2 \sigma_w^2$$

donde $n = n^{(\ell)}$ es el número de unidades de la capa ℓ .

Para preservar la varianza de las señales que se propagan durante el backward pass, debiésemos tener que $\sigma_w^2 \propto 1/n$ en vez de $\sigma_w^2 \propto 1/m$.

Inicialización Aleatoria - Varianza

- Un balance entre ambos objetivos se obtiene eligiendo $\sigma_w^2 \propto 1/((n + m)/2) = 2/(n + m)$ que genera la regla

$$W_{ij} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right)$$

donde m es la dimensionalidad de entrada de la capa y n es el número de unidades de la capa.

- Se utiliza también la versión normal de este criterio

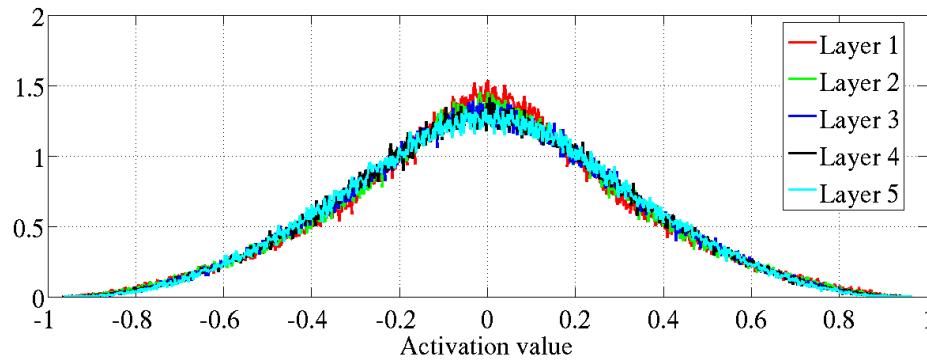
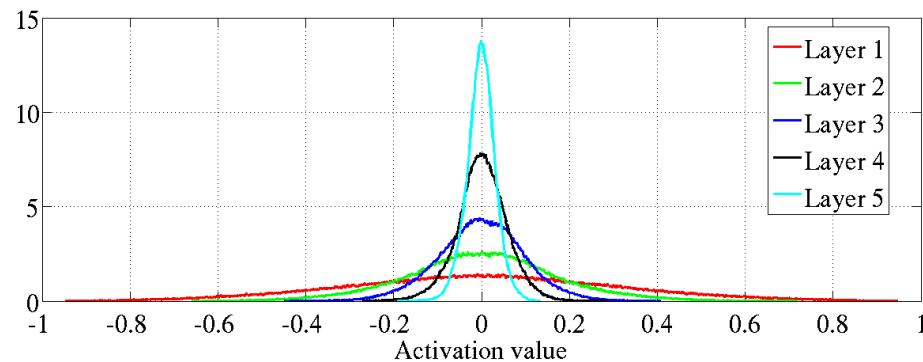
$$W_{ij} \sim \mathcal{N}(0, 2/(n + m))$$



Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Inicialización Aleatoria - Varianza

- Activaciones x capa usando el criterio clásico (arriba) versus el criterio propuesto (abajo). Todo al comienzo de entrenamiento.



Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Normalización

- La inicialización de pesos con los criterios mencionados anteriormente suele sincronizarse con una normalización de los datos de entrada (x) que permita obtener pre-activaciones centradas en 0 con varianza limitada.
- Si nuestra red acepta vectores de atributos $x \in \mathbb{R}^d$, la normalización estándar consiste en la siguiente transformación

$$\hat{x}_j = \frac{x_j - \mathbb{E}(x_j)}{\sigma_j} \quad \sigma_j^2 = \text{var}(x_j) = \mathbb{E}(x_j^2) - \mathbb{E}(x_j)^2$$

- Naturalmente las estadísticas involucradas se deben estimar desde el conjunto de ejemplos de entrenamiento

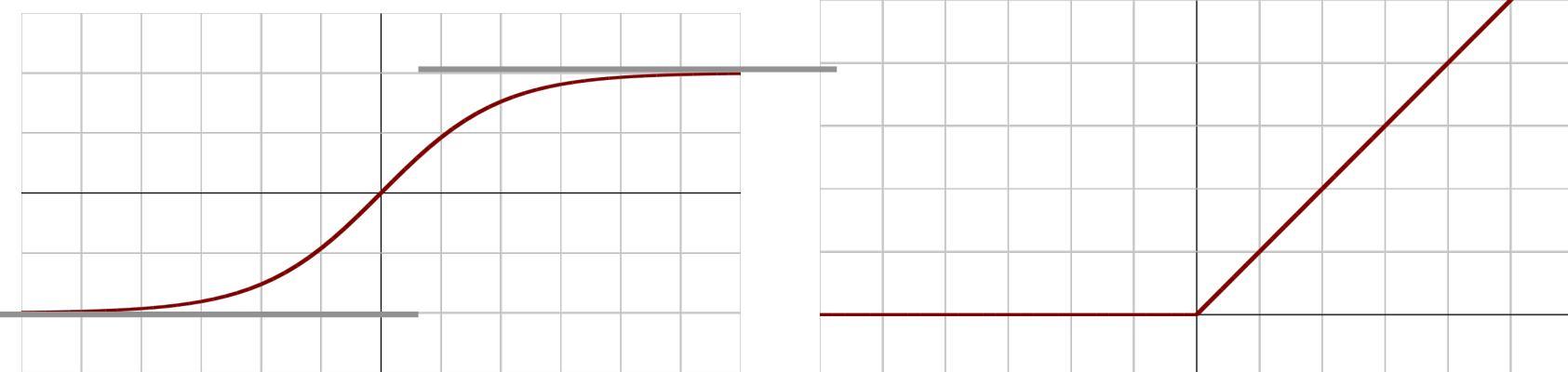
$$\mathbb{E}(x_j) \approx \bar{x}_j = \frac{1}{n} \sum_i x_j^{(i)}$$

$$\sigma_j^2 \approx S_j^2 = \frac{1}{n-1} \sum_i (x_j^{(i)} - \bar{x}_j)^2$$



Normalización

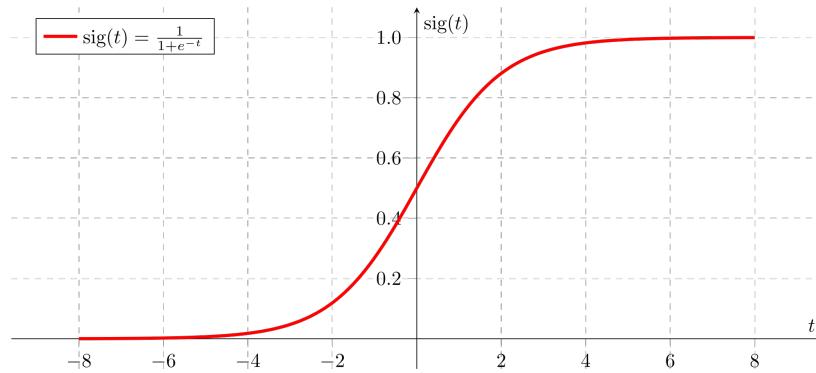
- Esta normalización (u otra similar) puede ser determinante en el éxito del entrenamiento, en gran parte, como mencionamos, porque la “sensibilidad” de las neuronas es máxima en torno a 0.



- **Problema:** A medida que se atraviesan las capas, no necesariamente se mantiene la normalización de los datos de entrada. Tanto la media como la varianza pueden alejarse de sus valores de referencia.

Normalización

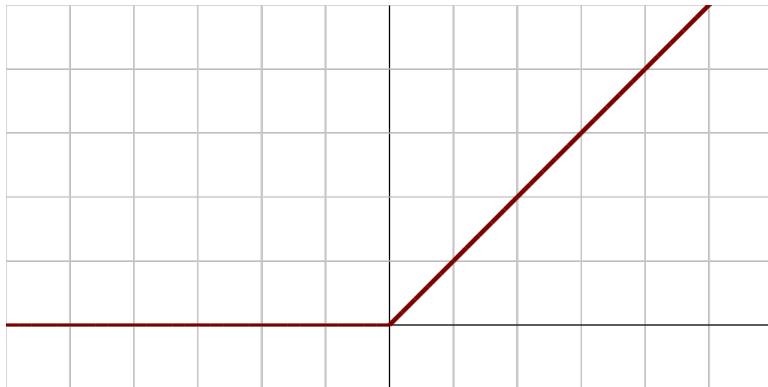
- En particular, muchas funciones de activación no preservan la media.
- Por ejemplo, si consideramos una unidad sigmoidal



$$p \sim \mathcal{N}(0, \sigma^2) \Rightarrow \mathbb{E}(\sigma(p)) \approx 1/2 \quad (\text{hasta al menos tercer orden})$$

Normalización

- En particular, muchas funciones de activación no preservan la media.
- Por ejemplo, si consideramos una rectificadora

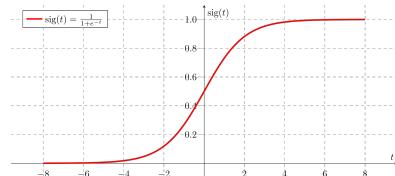


$$\begin{aligned} p \sim \mathcal{N}(0, \sigma^2) \Rightarrow \quad \mathbb{E}(g(p)) &\approx \int_{-\infty}^{\infty} f(p)g(p)dp \\ &= (2\pi\sigma^2)^{-1/2} \int_0^{\infty} pe^{-p^2/2\sigma^2} dp \\ &= (2\pi)^{-1/2}\sigma \int_0^{\infty} e^{-u} du = (2\pi)^{-1/2}\sigma \end{aligned}$$

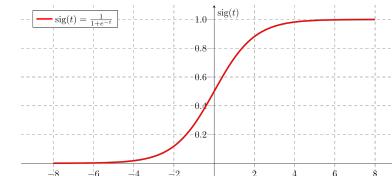
Normalización

- Este efecto, puede propagarse de capa en capa

$$\mathbb{E}(a^{(0)}) \approx 0$$



$$\mathbb{E}(a^{(1)}) \approx 1/2$$

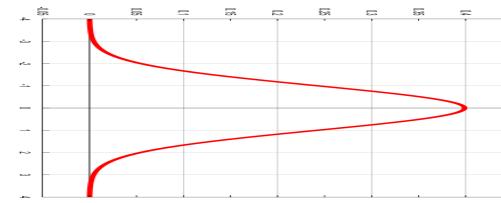
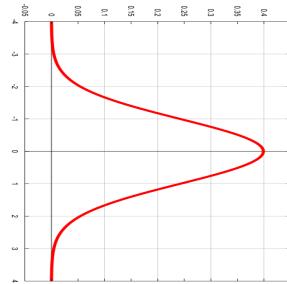
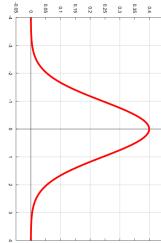


$$\mathbb{E}(a^{(2)}) \approx 0.62$$

- Algo similar ocurre con la varianza. El análisis que hicimos para obtener la inicialización de Glorot, se puede alejar bastante de la realidad cuando, a medida que progresá el entrenamiento, nos alejamos del régimen de linealidad (o de cercanía a 0) en cada unidad oculta.

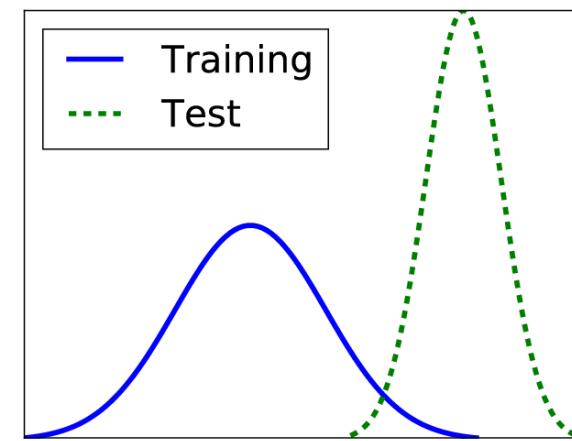
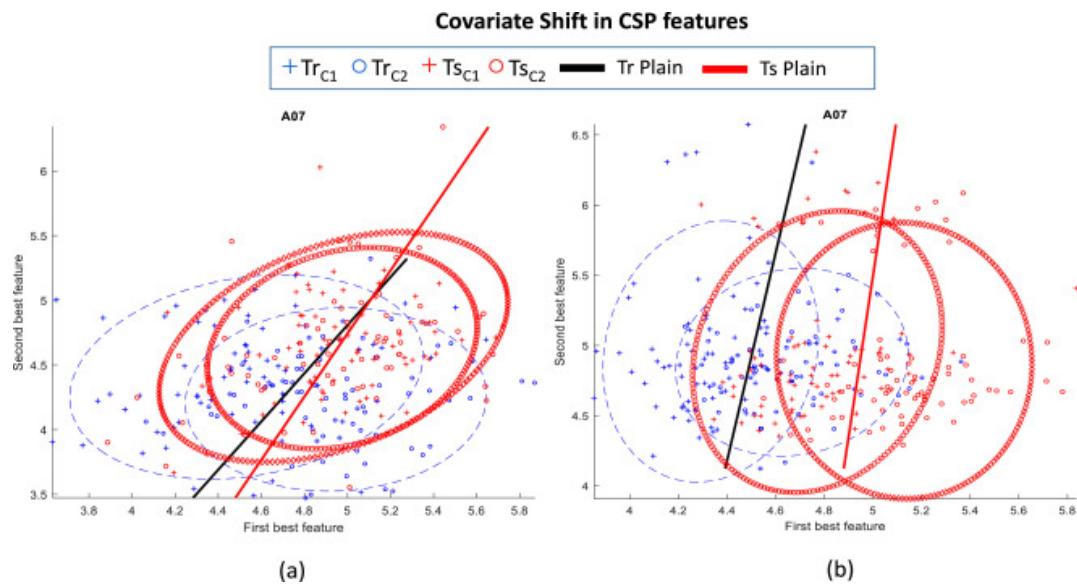
Normalización

- Sin embargo, cada capa oculta de la red, por muy profunda que sea, se ve afectada del mismo modo que la primera si los datos de entrada tienen una distribución muy alejada de la distribución de referencia para su función de activación



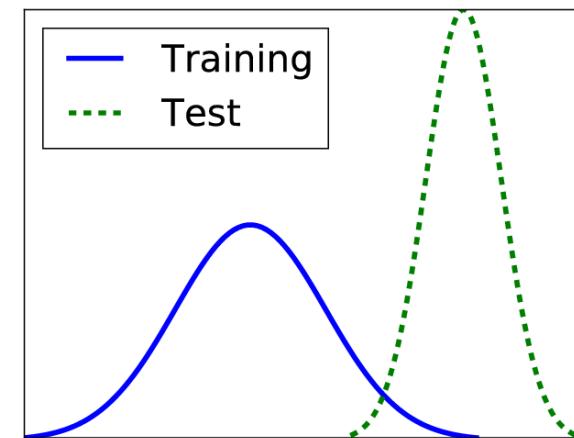
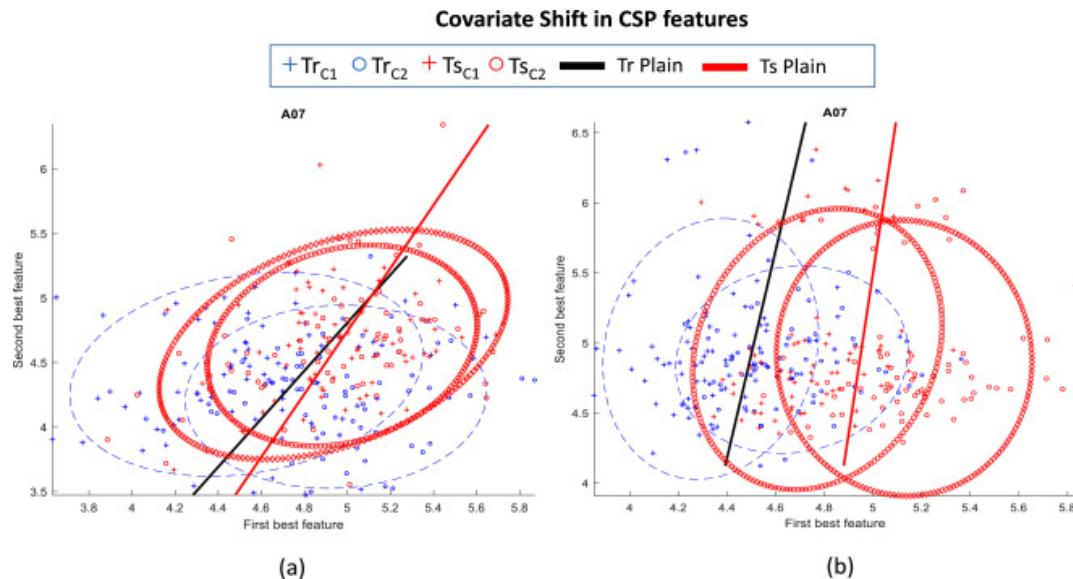
Internal Covariate Shift

- El cambio que se produce en la distribución de las activaciones de una capa tiene en realidad un efecto más profundo que simplemente sacarlas del rango de sensibilidad de la capa de sigue.
- En Machine Learning se denomina covariate shift al problema en que la distribución de los datos (x) de entrenamiento es diferente a la distribución de los datos (x) que el modelo tendrá que procesar en el futuro (test).



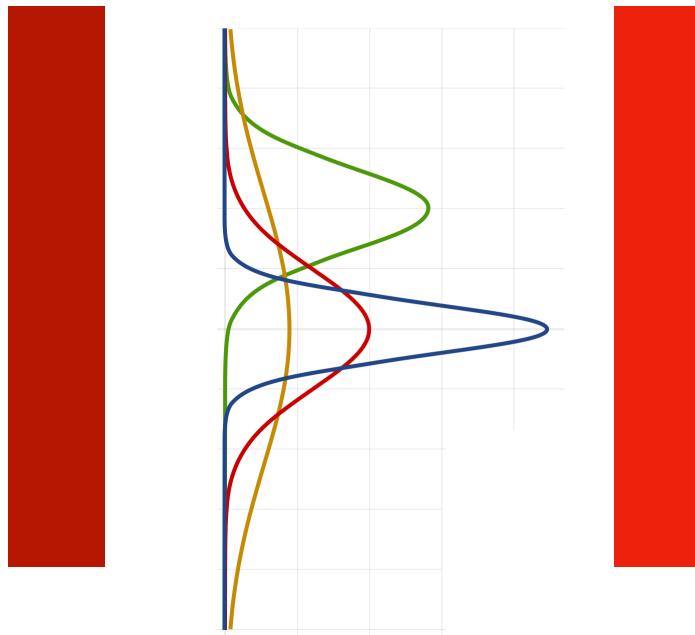
Internal Covariate Shift

- Dado que el conjunto de ejemplos sólo es útil porque nos permite predecir cómo serán los datos que tendremos que procesar en el futuro, la presencia de covariate shift es uno de los problemas más graves que podríamos enfrentar en ML (e imposible de resolver sin información adicional).



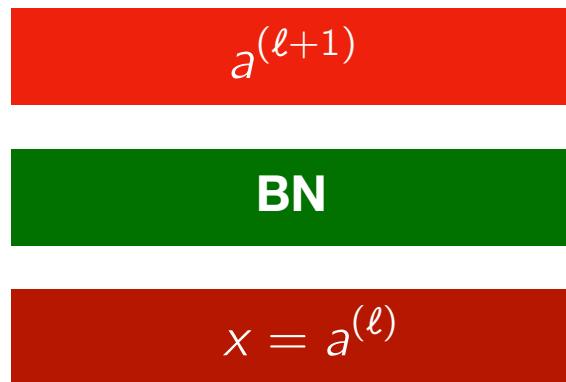
Internal Covariate Shift

- Bueno, lo que sucede a una capa oculta durante el entrenamiento de una red profunda es exactamente esto! Como la capa anterior cambia constantemente a medida que se entrena, las activaciones que le entrega a la capa siguiente cambian también de manera continua.



Batch Normalization

- Hacia 2015, Ioffe & Szegedy proponen Batch Normalization (BN), una capa que corrige explícitamente este problema.
- Si denotamos por x el output de la capa anterior a BN, la idea es recalculiar la normalización que típicamente se hace sobre los datos de entrada.



$$\hat{x}_j = \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j + \epsilon} \quad \forall j$$

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning*. 2015.

Batch Normalization

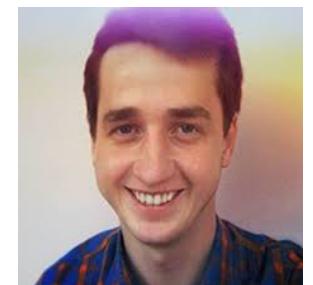
- Como la red se entrena mediante mini-batches (SGD), la media y la varianza **se estiman (recalculan) en cada mini-batch (no se entrenan vía gradiente)**

$$\hat{x}_j = \frac{x_j - \hat{\mu}_j^{(b)}}{\hat{\sigma}_j^{(b)} + \epsilon} \quad \forall j$$

$a^{(\ell+1)}$
BN
 $a^{(\ell)} =: x$

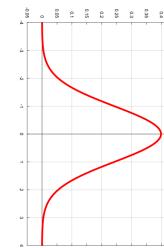
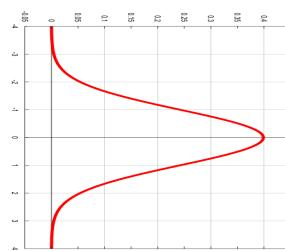
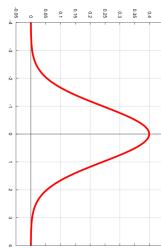
$$\hat{\mu}_j^{(b)} = \frac{1}{B} \sum_{i^*=1}^B x_j^{(i^*)} \quad \hat{\sigma}_j^{(b)} = \frac{1}{B} \sum_{i^*=1}^B (x_j^{(i^*)} - \hat{\mu}_j^{(b)})^2$$

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning*. 2015.



Normalización

- Sin embargo, cada capa oculta de la red, por muy profunda que sea, se ve afectada del mismo modo que la primera si los datos de entrada tienen una distribución muy alejada de la distribución de referencia para su función de activación



Batch Normalization

- Para permitir a la red “recuperar” la representación original que de la capa que estamos alterando, se agregan dos parámetros entrenables (por dimensión)

$a^{(\ell+1)}$

$\mathbf{y} = \mathbf{BN}(\mathbf{x})$

$$y = \gamma_j \hat{x}_j + \beta_j$$

$$\hat{x}_j = \frac{x_j - \hat{\mu}_j^{(b)}}{\hat{\sigma}_j^{(b)} + \epsilon} \quad \forall j$$

$a^{(\ell)} =: x$

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning*. 2015.



Batch Normalization

- Cuando termina el entrenamiento, los parámetros γ_j, β_j se congelan, como los demás parámetros entrenables. La media y la varianza también se fijan, pero los autores sugieren re-calcular estos parámetros tomando un “valor esperado sobre mini-batches”.

$a^{(\ell+1)}$

$y = \text{BN}(x)$

$a^{(\ell)} =: x$

$$y = \gamma_j \hat{x}_j + \beta_j$$

$$\hat{x}_j = \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j + \epsilon} \quad \forall j$$

$$\hat{\mu}_j = \frac{1}{N_b} \sum_{b=1}^{N_b} \hat{\mu}_j^{(b)}$$

$$\hat{\sigma}_j^2 = \frac{b}{b-1} \sum_{b=1}^{N_b} \hat{\sigma}_j^{(b)2}$$

(*) La corrección en el estimador de la varianza se usa porque después del entrenamiento, interesa corregir las activaciones que se verificarán sobre toda la población. En entrenamiento se quiere normalizar el mini-batch.

Batch Normalization

- Alternativamente, se pueden mantener medias móviles durante el entrenamiento (se evita el cálculo final).

$a^{(\ell+1)}$

$\mathbf{y} = \mathbf{BN}(\mathbf{x})$

$$\begin{aligned}\hat{\mu}_j &\leftarrow \lambda \hat{\mu}_j + (1 - \lambda) \hat{\mu}_j^{(b)} \\ \hat{\sigma}_j^2 &\leftarrow \lambda \hat{\sigma}_j^2 + (1 - \lambda) \frac{b}{b-1} \hat{\sigma}_j^{(b)2}\end{aligned}$$

$a^{(\ell)} =: x$

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning*. 2015.



Batch Normalization

- Como las operaciones de la capa son todas lineales, no es difícil calcular las derivadas que permiten completar el back-propagation por la capa.

$$\frac{\partial L}{\partial \hat{x}_j^{(i*)}} = \gamma_j \frac{\partial L}{\partial y_j^{(i*)}}$$

$$\frac{\partial L}{\partial \hat{\sigma}_j^2} = - \sum_{i^*=1}^B \frac{\partial L}{\partial \hat{x}_j^{(i*)}} (x_j^{(i*)} - \mu_j) \frac{1}{2} (\hat{\sigma}_j^2 + \epsilon)^{3/2}$$

$$\frac{\partial L}{\partial \hat{\mu}_j} = \left(\sum_{i^*=1}^B \frac{\partial L}{\partial \hat{x}_j^{(i*)}} \frac{-1}{\sqrt{\hat{\sigma}_j^2 + \epsilon}} \right) - \frac{\partial L}{\partial \hat{\sigma}_j^2} \frac{\sum_{i^*}^B 2(\hat{x}_j^{(i*)} - \hat{\mu}_j)}{B}$$

$$\boxed{\frac{\partial L}{\partial x_j^{(i*)}} = \frac{\partial L}{\partial \hat{x}_j^{(i*)}} \frac{1}{\sqrt{\hat{\sigma}_j^2 + \epsilon}} + \frac{\partial L}{\partial \hat{\sigma}_j^2} \frac{2(\hat{x}_j^{(i*)} - \hat{\mu}_j)}{B} + \frac{\partial L}{\partial \hat{\mu}_j} \frac{1}{B}}$$

Esta se propaga hacia la capa anterior

$$\frac{\partial L}{\partial \gamma_j} = \sum_{i^*=1}^B \frac{\partial L}{\partial y_j^{(i*)}} \hat{x}_j^{(i*)}$$

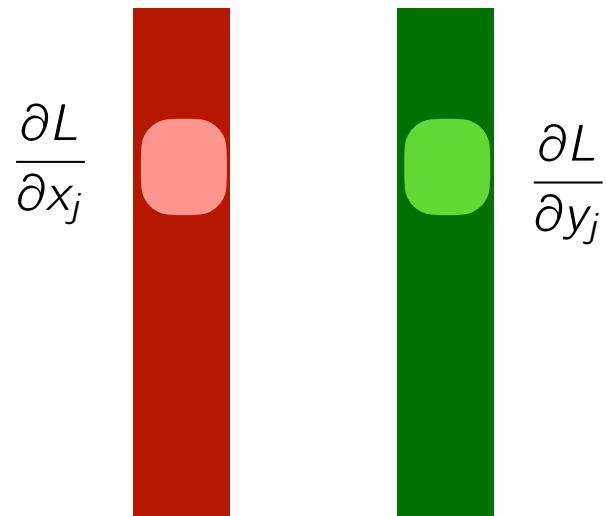
$$\frac{\partial L}{\partial \beta_j} = \sum_{i^*=1}^B \frac{\partial L}{\partial y_j^{(i*)}}$$

Batch Normalization como Regularizador

- Este cálculo nos muestra también que, además de normalizar la escala de las activaciones de la capa anterior, BN re-escala los gradientes justo antes de entrar a esa capa durante el backward pass

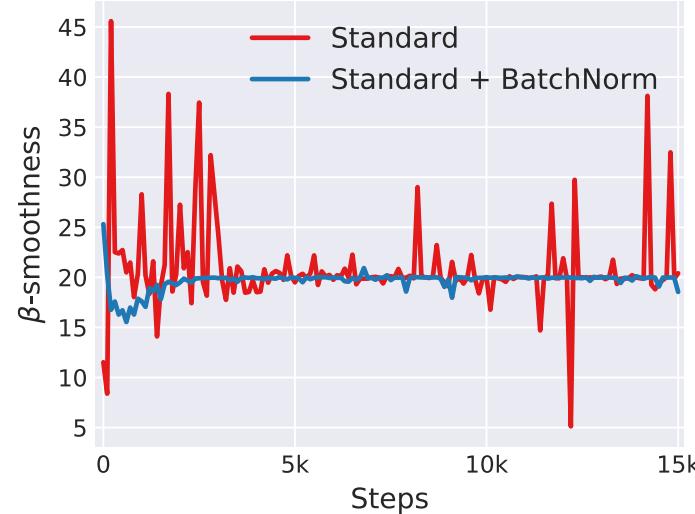
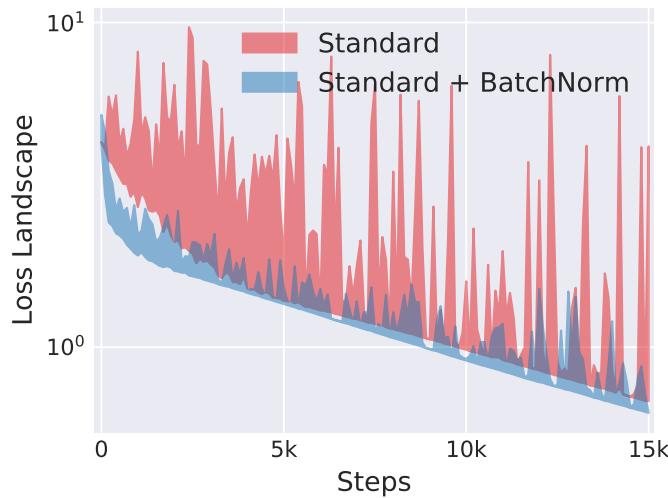
$$\frac{\partial L}{\partial \hat{x}_j^{(i*)}} = \gamma_j \frac{\partial L}{\partial y_j^{(i*)}}$$

$$\frac{\partial L}{\partial x_j^{(i*)}} = \frac{\partial L}{\partial \hat{x}_j^{(i*)}} \frac{1}{\sqrt{\hat{\sigma}_j^2 + \epsilon}} + \frac{\partial L}{\partial \hat{\sigma}_j^2} \frac{2(\hat{x}_j^{(i*)} - \hat{\mu}_j)}{B} + \frac{\partial L}{\partial \hat{\mu}_j} \frac{1}{B}$$



Batch Normalization como “Estabilizador”

- Como argumenta recientemente Shibani, esto podría explicar el efecto “estabilizador” de la loss que se observa en la práctica: gradientes más suaves generan un progreso más suave del optimizador durante el entrenamiento.



Una f.o. es β -smooth si sus gradientes son β -Lipschitz, es decir, el cambio máximo que sufren relativamente al cambio en el argumento es β . En este caso, se calculó cuánto cambiaban los gradientes moviéndose en la dirección indicada por ellos durante el entrenamiento.

Santurkar, Shibani, et al. "How does batch normalization help optimization?." *Advances in Neural Information Processing Systems*. 2018.



Batch Normalization como “Estabilizador”

- Recordemos que en el análisis de convergencia de BP estocástico nos aparecía un término que acotamos usando el segundo momento de los gradientes aproximados que tomamos en cada iteración
- Si los gradientes son más estables, nos debiésemos esperar que esa constante sea menor

$$E(\mathbf{w}^{(t+1)}) \leq E(\mathbf{w}^{(t)}) + \nabla E(\mathbf{w}^{(t)})^T (\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}) + \frac{L}{2} \|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|^2$$

$$= E(\mathbf{w}^{(t)}) - \eta_t \nabla E(\mathbf{w}^{(t)})^T \tilde{\nabla} E(\mathbf{w}^{(t)}) + \frac{L \eta_t^2}{2} \|\tilde{\nabla} E(\mathbf{w}^{(t)})\|^2$$

**1: reducción
de la f.o.**

**2: aumento
de la f.o.**

Si $1 > 2$, el paso empeora la solución actual.

Batch Normalization como “Estabilizador”

- Shibani muestra explícitamente que es así!

Theorem 4.4 (Minimax bound on weight-space Lipschitzness). *For a BatchNorm network with loss $\widehat{\mathcal{L}}$ and an identical non-BN network (with identical loss \mathcal{L}), if*

$$g_j = \max_{\|X\| \leq \lambda} \|\nabla_W \mathcal{L}\|^2, \quad \hat{g}_j = \max_{\|X\| \leq \lambda} \left\| \nabla_W \widehat{\mathcal{L}} \right\|^2 \implies \hat{g}_j \leq \frac{\gamma^2}{\sigma_j^2} \left(g_j^2 - m\mu_{g_j}^2 - \lambda^2 \langle \nabla_{y_j} \mathcal{L}, \hat{y}_j \rangle^2 \right).$$

- Se demuestra también que los gradientes y la Hessiana se alinean mejor

Theorem 4.2 (The effect of BN to smoothness). *Let $\hat{g}_j = \nabla_{y_j} \mathcal{L}$ and $H_{jj} = \frac{\partial \mathcal{L}}{\partial y_j \partial y_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively. Then*

$$\left(\nabla_{y_j} \widehat{\mathcal{L}} \right)^\top \frac{\partial \widehat{\mathcal{L}}}{\partial y_j \partial y_j} \left(\nabla_{y_j} \widehat{\mathcal{L}} \right) \leq \frac{\gamma^2}{\sigma^2} \left(\frac{\partial \widehat{\mathcal{L}}}{\partial y_j} \right)^\top H_{jj} \left(\frac{\partial \widehat{\mathcal{L}}}{\partial y_j} \right) - \frac{\gamma}{m\sigma^2} \langle \hat{g}_j, \hat{y}_j \rangle \left\| \frac{\partial \widehat{\mathcal{L}}}{\partial y_j} \right\|^2$$

Santurkar, Shibani, et al. "How does batch normalization help optimization?." *Advances in Neural Information Processing Systems*. 2018.



Batch Normalization como “Estabilizador”

- Esto significa que los gradientes son “localmente más predictivos” gracias a BN, es decir, es más seguro que avanzando en la dirección que indican como update para una capa logremos reducir el valor del objetivo

Theorem 4.2 (The effect of BN to smoothness). *Let $\hat{\mathbf{g}}_j = \nabla_{\mathbf{y}_j} \mathcal{L}$ and $\mathbf{H}_{jj} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}_j \partial \mathbf{y}_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively. Then*

$$\left(\nabla_{\mathbf{y}_j} \hat{\mathcal{L}} \right)^T \frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{y}_j \partial \mathbf{y}_j} \left(\nabla_{\mathbf{y}_j} \hat{\mathcal{L}} \right) \leq \frac{\gamma^2}{\sigma^2} \left(\frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{y}_j} \right)^T \mathbf{H}_{jj} \left(\frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{y}_j} \right) - \frac{\gamma}{m\sigma^2} \langle \hat{\mathbf{g}}_j, \hat{\mathbf{y}}_j \rangle \left\| \frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{y}_j} \right\|^2$$

$$E(\theta^{(t+1)}) \approx E(\theta^{(t)}) + \nabla_{\theta} E^T \left(\theta^{(t+1)} - \theta^{(t)} \right) + \left(\theta^{(t+1)} - \frac{1}{2}\theta^{(t)} \right)^T H(\theta^{(t)}) \left(\theta^{(t+1)} - \theta^{(t)} \right)$$
$$E(\theta^{(t+1)}) - E(\theta^{(t)}) \approx \eta \nabla_{\theta} E^T d^{(t)} + \frac{\eta^2}{2} d^{(t)T} H d^{(t)}$$

Santurkar, Shibani, et al. "How does batch normalization help optimization?." *Advances in Neural Information Processing Systems*. 2018.



Batch Normalization como “Estabilizador”

- Recordemos que si en la iteración t avanzamos en una dirección $d^{(t)}$, el valor del objetivo se puede aproximar, a segundo orden, como

$$E(\theta^{(t+1)}) \approx E(\theta^{(t)}) + \nabla_{\theta} E^T (\theta^{(t+1)} - \theta^{(t)}) + \left(\theta^{(t+1)} - \frac{1}{2}\theta^{(t)} \right)^T H(\theta^{(t)}) (\theta^{(t+1)} - \theta^{(t)})$$

$$E(\theta^{(t+1)}) - E(\theta^{(t)}) \approx n \nabla_{\theta} E^T d^{(t)} + \frac{\eta^2}{2} d^{(t)T} H d^{(t)}$$

**1: reducción
de la f.o.**

**2: aumento
de la f.o.**

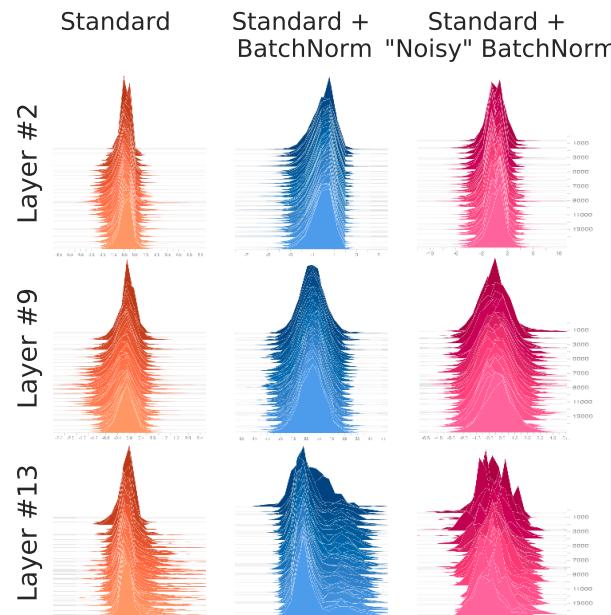
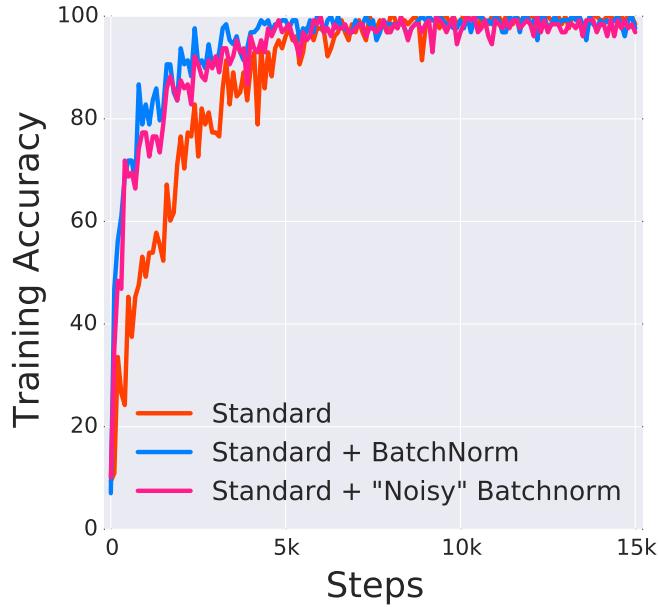
- El teorema de Shibani muestra que usando BP, 2 está acotado

Theorem 4.2 (The effect of BN to smoothness). *Let $\hat{g}_j = \nabla_{y_j} \mathcal{L}$ and $H_{jj} = \frac{\partial \mathcal{L}}{\partial y_j \partial y_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively. Then*

$$\left(\nabla_{y_j} \hat{\mathcal{L}} \right)^T \frac{\partial \hat{\mathcal{L}}}{\partial y_j \partial y_j} \left(\nabla_{y_j} \hat{\mathcal{L}} \right) \leq \frac{\gamma^2}{\sigma^2} \left(\frac{\partial \hat{\mathcal{L}}}{\partial y_j} \right)^T H_{jj} \left(\frac{\partial \hat{\mathcal{L}}}{\partial y_j} \right) - \frac{\gamma}{m\sigma^2} \langle \hat{g}_j, \hat{y}_j \rangle \left\| \frac{\partial \hat{\mathcal{L}}}{\partial y_j} \right\|^2$$

Batch Normalization como Regularizador

- La autora argumenta también que la explicación del covariate shift interno como justificación de efecto **regularizador** BN es limitada, porque es común que la técnica mantenga o empeore la deriva distribucional interna y que aún así mejore el error de predicción

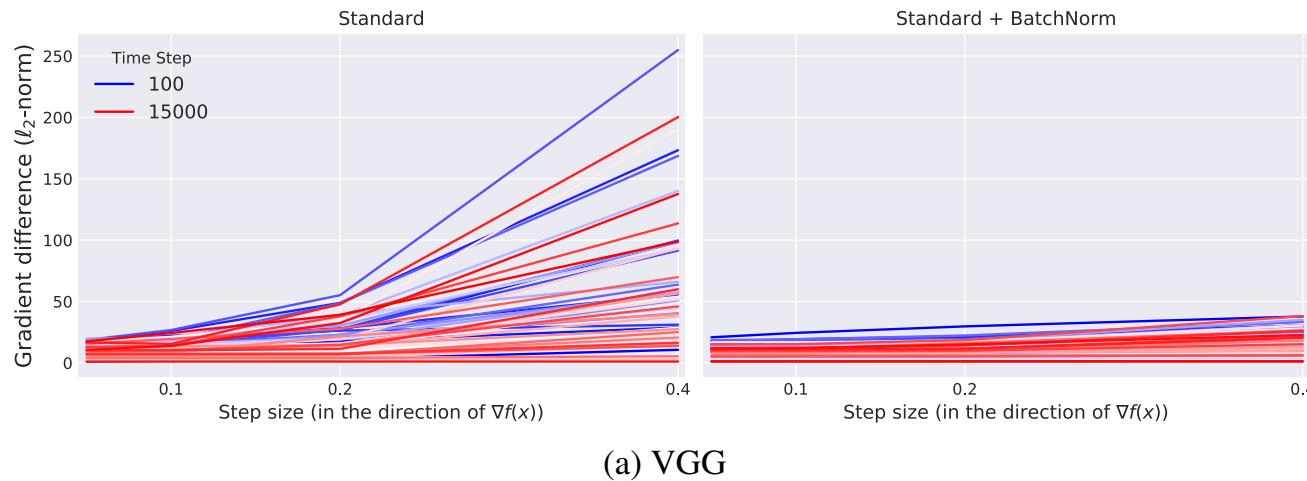


Noisy version consiste en sumar ruido blanco a la salida de la capa BN.



Batch Normalization como Regularizador

- Su explicación para el hecho de que BN logre mejorar el error de predicción es que la mayor estabilidad de los gradientes permite utilizar tasas de aprendizaje significativamente más grandes (*): esto permite explorar mejor el espacio de búsqueda sin estancarse en un punto estacionario poco prometedor (*)



Cambio de los gradientes cuando nos movemos en la dirección que indican.



Santurkar, Shibani, et al. "How does batch normalization help optimization?." *Advances in Neural Information Processing Systems*. 2018.

Batch Normalization como Regularizador

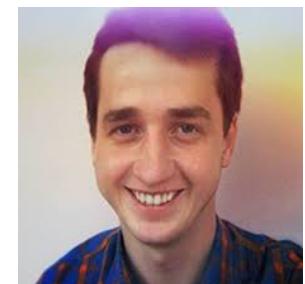
- Todos estos argumentos son también compatibles con una explicación/nota a margen en el paper original de 2015 en que se menciona que gracias a BN, la influencia de **un** ejemplo en la optimización/aprendizaje se ve reducida, reduciendo la probabilidad de que la red lo memorice.

3.4 Batch Normalization regularizes the model

When training with Batch Normalization, a training example is seen in conjunction with other examples in the mini-batch, and the training network no longer producing deterministic values for a given training example. In our experiments, we found this effect to be advantageous to the generalization of the network. Whereas Dropout (Srivastava et al., 2014) is typically used to reduce overfitting, in a batch-normalized network we found that it can be either removed or reduced in strength.

4 Experiments

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning*. 2015.



Batch Normalization como Regularizador

- Todos estos argumentos son también compatibles con explicaciones más intuitivas sobre el efecto regularizador de BN: a depender de sólo 2 parámetros entrenables, la capa BN reduce las interacciones entre dos capas sucesivas de la red haciéndolas más fáciles de aprender.

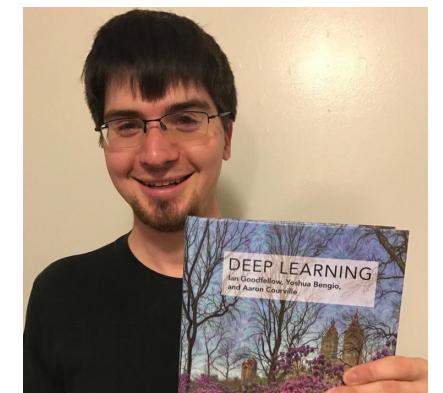
$$E(\theta^{(t+1)}) \approx E(\theta^{(t)}) + \nabla_{\theta} E^T \left(\theta^{(t+1)} - \theta^{(t)} \right) + \left(\theta^{(t+1)} - \frac{1}{2}\theta^{(t)} \right)^T H(\theta^{(t)}) \left(\theta^{(t+1)} - \theta^{(t)} \right)$$

$$E(\theta^{(t+1)}) - E(\theta^{(t)}) \approx \eta \nabla_{\theta} E^T d^{(t)} + \frac{\eta^2}{2} d^{(t)T} H d^{(t)}$$

$a^{(\ell+1)}$ $S^{(\ell+1)}xS^{(\ell)}$ Params

BN $2S^{(\ell)}$ Params

$a^{(\ell)} =: x$ $S^{(\ell)}xS^{(\ell-1)}$ Params



Batch Normalization en Redes Convolucionales

- Como hemos mencionado, BN se ha demostrado mucho más efectivo que Dropout en redes convolucionales.
- La versión original de BN sobre este tipo de capas prevé la normalización de un canal calculando las estadísticas sobre todas las activaciones de ese canal. En el caso 2D

$$\mu, \sigma, \gamma, \beta$$

$$1 \times 1 \times 1 \times C$$

$$X$$

$$B \times H \times W \times C$$

Batch Normalization en Redes Convolucionales

- Variantes de este estándar muestran funcionan igualmente bien

$$\mu, \sigma, \gamma, \beta$$

$$1 \times 1 \times 1 \times C$$

$$X$$

$$B \times H \times W \times C$$

$$\mu, \sigma, \gamma, \beta$$

$$B \times 1 \times 1 \times C$$

$$X$$

$$B \times H \times W \times C$$

$$\mu, \sigma, \gamma, \beta$$

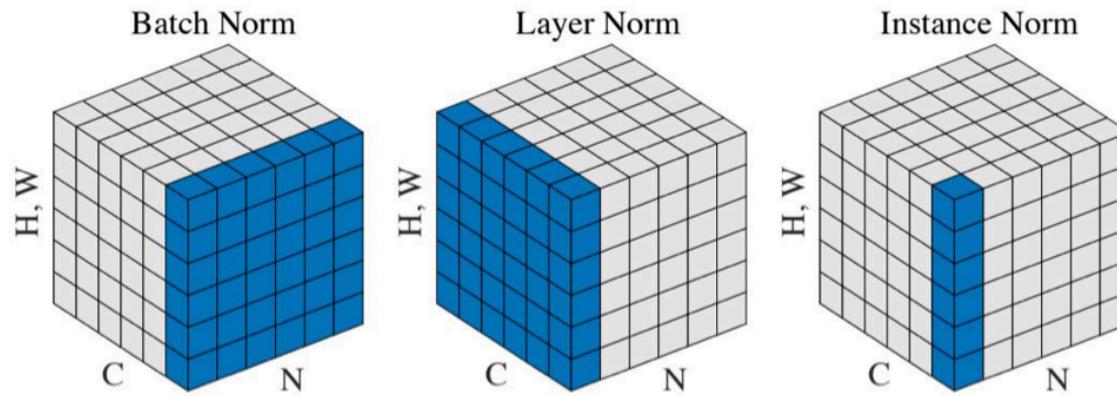
$$B \times 1 \times 1 \times 1$$

$$X$$

$$B \times H \times W \times C$$

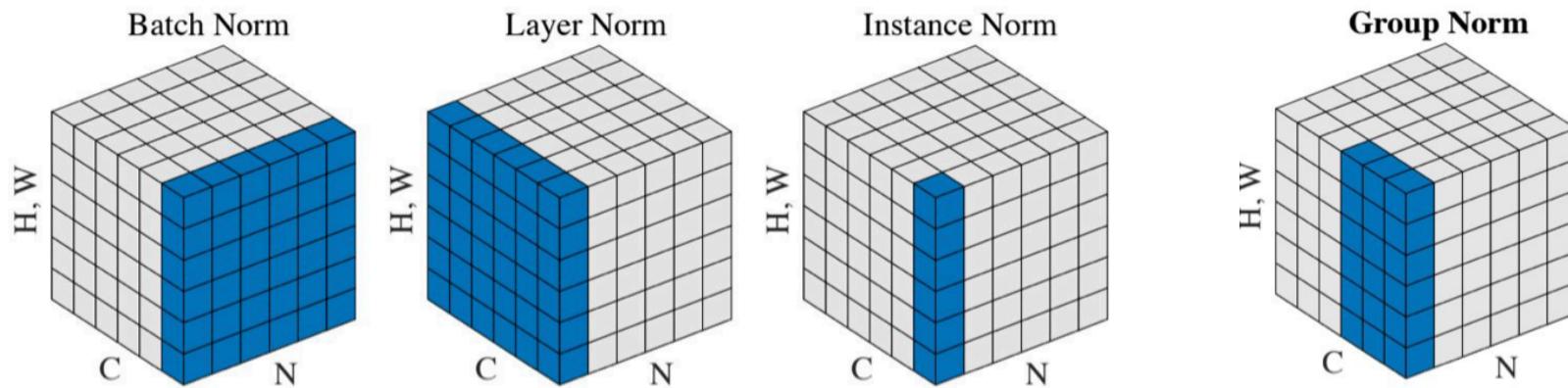
Batch Normalization en Redes Convolucionales

- Variantes de este estándar muestran funcionan igualmente bien



Batch Normalization en Redes Convolucionales

- Recientemente Wu ha propuesto normalizar de manera independiente del batch size usando trozos de activaciones. Esto permite usar batch sizes más pequeños sin comprometer la calidad de las estadísticas requeridas por BN.



Wu, Yuxin, and Kaiming He. "Group normalization." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.



Antes o Después de la No-Linealidad?

BN

Non-Linear Elementwise

Non-Linear Elementwise

BN

Linear Operation

Linear Operation

- Los autores de BN recomiendan aplicar la normalización justo antes de entrar a la no-linealidad (la operación lineal hace que las pre-activaciones sean más gausianas). Sin embargo, muchos autores han reportado mejores resultados aplicándola después.



Entonces ...

- Las técnicas de inicialización de los pesos de la red buscan (i) romper las simetrías naturales de la red reduciendo la co-adaptación de unidades, (ii) poner las activaciones en el rango de sensibilidad de las no-linealidades y (iii) ayudar a estabilizar los gradientes durante las fases iniciales de entrenamiento. Estos objetivos se relacionan con la capacidad de generalización de la red.
- Estos mecanismos se suelen acompañar de una normalización de los datos de entrada a la red. Iterando esta idea en profundidad aparece Batch Normalization (BN), una técnica muy efectiva para estabilizar el entrenamiento y reducir la tendencia de un modelo profundo a overfitting.
- Si bien la explicación clásica de BN se centra en la mayor estabilidad de las activaciones, la investigación más reciente sugiere que su efecto más importante es sobre los gradientes que se propagan durante el backward pass.

