Redes Neuronales

Matias Vera - Juan Zuloaga - Lautaro Estienne

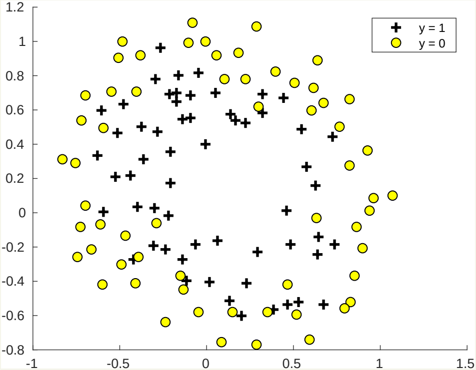
Centro de Simulación Computacional para Aplicaciones Tecnológicas

Agenda

1. Redes Neuronales como aproximador universal
2. Deep World
3. ¿Cómo entrenar una deep net?

Las soluciones lineales pueden ser insuficientes

¿Como reducir el riesgo empírico?



Aproximación Universal

Teorema

Sea $\mathcal{A} \subset \mathbb{R}^m$ un conjunto cerrado y acotado y sea $g : \mathbb{R} \to \mathbb{R}$. Para cada función continua $f : \mathcal{A} \to \mathbb{R}$ y $\epsilon > 0$ existe un $n \in \mathbb{N}$ y $\{\mathrm{w}_i, b_i, \alpha_i\}_{i=1}^n$ con $\mathrm{w}_i \in \mathbb{R}^m$ y $b_i, \alpha_i \in \mathbb{R}$ para todo $i = 1, \cdots, n$ tales que

$$\left| f(\mathrm{x}) - \sum_{i=1}^{n} \alpha_i g(\mathrm{w}_i^T \mathrm{x} + b_i) \right| < \epsilon \qquad \forall\, \mathrm{x} \in \mathcal{A}$$

si y solo si g no es un polinomio.

Necesito funciones no lineales!

Leshno and Schocken 1993: "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

Aproximación Universal

Teorema

Sea $\mathcal{A} \subset \mathbb{R}^m$ un conjunto cerrado y acotado y sea $g : \mathbb{R} \to \mathbb{R}$. Para cada función continua $f : \mathcal{A} \to \mathbb{R}$ y $\epsilon > 0$ existe un $n \in \mathbb{N}$ y $\{\mathrm{w}_i, b_i, \alpha_i\}_{i=1}^n$ con $\mathrm{w}_i \in \mathbb{R}^m$ y $b_i, \alpha_i \in \mathbb{R}$ para todo $i = 1, \cdots, n$ tales que

$$\left| f(\mathrm{x}) - \sum_{i=1}^{n} \alpha_i g(\mathrm{w}_i^T \mathrm{x} + b_i) \right| < \epsilon \qquad \forall\, \mathrm{x} \in \mathcal{A}$$

si y solo si g no es un polinomio.

Necesito funciones no lineales!

Regresión

$$\mathrm{x} \longrightarrow \boxed{\mathrm{W}\mathrm{x} + \mathrm{b}} \xrightarrow{\mathrm{z}^{[1]}} \boxed{g(\mathrm{z}^{[1]})} \xrightarrow{\mathrm{h}} \boxed{\mathrm{A}\mathrm{h} + \mathrm{c}} \longrightarrow \hat{y}$$

Leshno and Schocken 1993: "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

Aproximación Universal

Teorema

Sea $\mathcal{A} \subset \mathbb{R}^m$ un conjunto cerrado y acotado y sea $g : \mathbb{R} \to \mathbb{R}$. Para cada función continua $f : \mathcal{A} \to \mathbb{R}$ y $\epsilon > 0$ existe un $n \in \mathbb{N}$ y $\{\mathrm{w}_i, b_i, \alpha_i\}_{i=1}^n$ con $\mathrm{w}_i \in \mathbb{R}^m$ y $b_i, \alpha_i \in \mathbb{R}$ para todo $i = 1, \cdots, n$ tales que

$$\left| f(\mathrm{x}) - \sum_{i=1}^{n} \alpha_i g(\mathrm{w}_i^T \mathrm{x} + b_i) \right| < \epsilon \qquad \forall\, \mathrm{x} \in \mathcal{A}$$

si y solo si g no es un polinomio.

Necesito funciones no lineales!

Clasificación

$$\mathrm{x} \longrightarrow \boxed{\mathrm{Wx} + \mathrm{b}} \xrightarrow{\mathrm{z}^{[1]}} \boxed{g(\mathrm{z}^{[1]})} \xrightarrow{\mathrm{h}} \boxed{\mathrm{Ah} + \mathrm{c}} \xrightarrow{\mathrm{z}^{[2]}} \boxed{\frac{e^{\mathrm{z}^{[2]}}}{1^T e^{\mathrm{z}^{[2]}}}} \longrightarrow \mathrm{f}(\mathrm{x})$$

Leshno and Schocken 1993: "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

Aproximación Universal

Teorema

Sea $\mathcal{A} \subset \mathbb{R}^m$ un conjunto cerrado y acotado y sea $g : \mathbb{R} \to \mathbb{R}$. Para cada función continua $f : \mathcal{A} \to \mathbb{R}$ y $\epsilon > 0$ existe un $n \in \mathbb{N}$ y $\{\mathrm{w}_i, b_i, \alpha_i\}_{i=1}^n$ con $\mathrm{w}_i \in \mathbb{R}^m$ y $b_i, \alpha_i \in \mathbb{R}$ para todo $i = 1, \cdots, n$ tales que

$$\left| f(\mathrm{x}) - \sum_{i=1}^{n} \alpha_i g(\mathrm{w}_i^T \mathrm{x} + b_i) \right| < \epsilon \qquad \forall\, \mathrm{x} \in \mathcal{A}$$

si y solo si g no es un polinomio.

Necesito funciones no lineales!

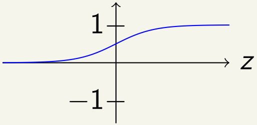
Autoencoder

$$\mathrm{x} \longrightarrow \boxed{\mathrm{Wx} + \mathrm{b}} \xrightarrow{\mathrm{z}^{[1]}} \boxed{g(\mathrm{z}^{[1]})} \xrightarrow{\mathrm{h}} \boxed{\mathrm{Ah} + \mathrm{c}} \longrightarrow \hat{\mathrm{x}}$$

Leshno and Schocken 1993: "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

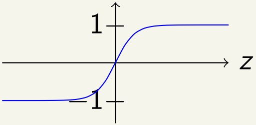
Funciones de Activación g

$$g(z) = \frac{1}{1+e^{-z}}$$



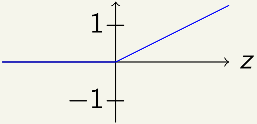
Sigmoide

$$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$



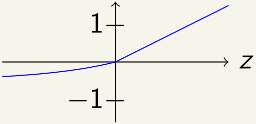
Tangente hiperbólica

$$g(z) = \max\{z, 0\}$$



ReLU

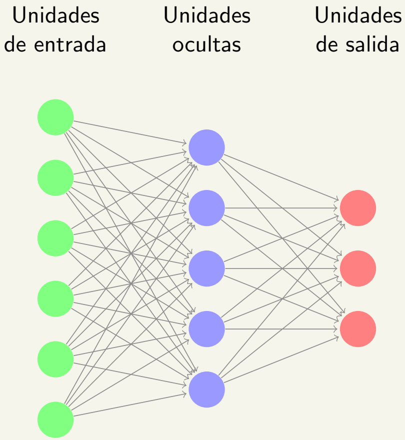
$$g(z) = \begin{cases} z & z \geq 0 \\ \alpha(e^{z} - 1) & z < 0 \end{cases}$$



$\alpha-$ELU

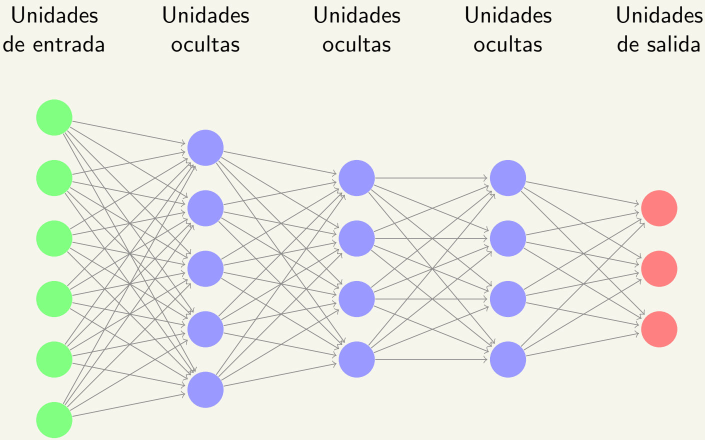
Agenda

Deep Learning



Red neuronal básica

Deep Learning



Deep Learning Architecture

¿Por que muchos layers?

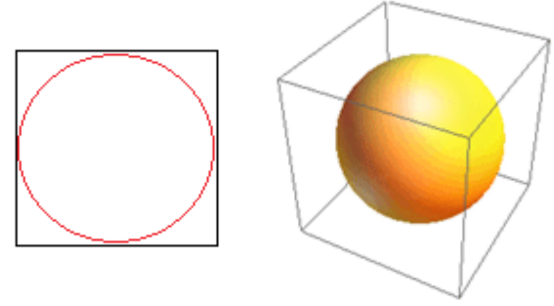
Los teoremas de aproximación universal nos hablan de la existencia de parámetros, no como elegirlos ni cuantas unidades ocultas usar. Algunas ventajas del deep-world:

- Las redes profundas suelen ser más eficientes en la etapa del entrenamiento y tienen mejores propiedades de generalización.
- La cantidad de unidades ocultas necesarias para aproximar suele decaer exponencialmente con la profundidad de la red.
- En algunos casos se puede demostrar que la cantidad de unidades ocultas necesarias crece exponencialmente con la dimensión de la entrada.

Características del Deep-world

- Big Data (A partir de 30K o 40K muestras)
- Alta dimensionalidad de la entrada (mayor a 10)
- Cantidad de parámetros a entrenar mayor (o al menos comparable) que la cantidad de muestras.
- Suficientes hiperparámetros para considerar hacer un barrido en grilla una mala opción.

La maldición de la dimensionalidad



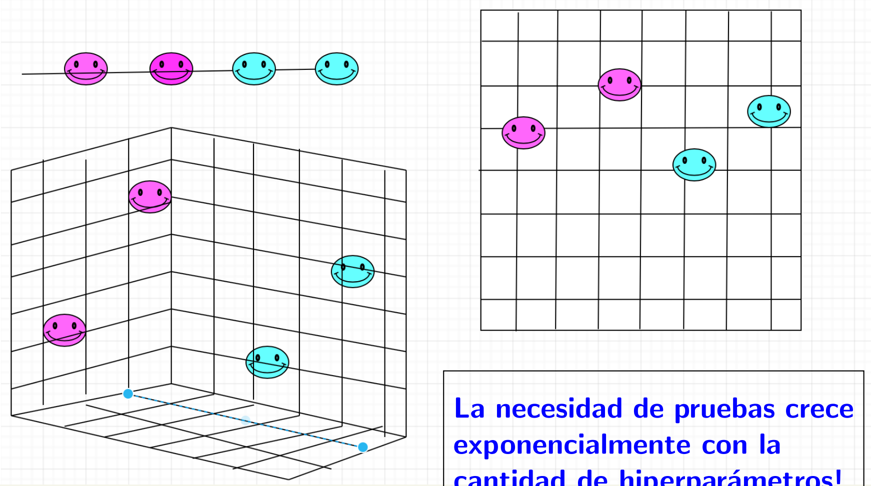
- 2d: $\frac{\pi r^2}{(2r)^2} \approx 78.5\%$
- 3d: $\frac{\frac{4}{3}\pi r^3}{(2r)^3} \approx 52.3\%$
- 10d: $\frac{\frac{r^{10}}{5!}\pi^5}{(2r)^{10}} \approx 0.25\%$

En grandes dimensiones:

- Los puntos están muy lejos.
- Las estructuras son muy sparce.
- La distancia euclidea no es buena métrica.
- La "necesidad" de muestras crece exponencialmente con la dimensión.
- La cantidad de mínimos locales crece exponencialmente con la dimensión.

La maldición de la dimensionalidad

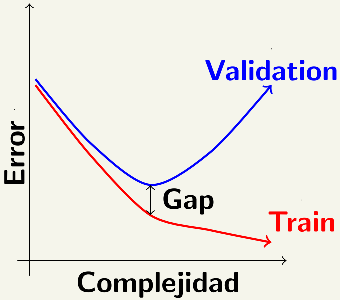
La maldición también aplica a los hiperparámetros



La necesidad de pruebas crece exponencialmente con la cantidad de hiperparámetros!

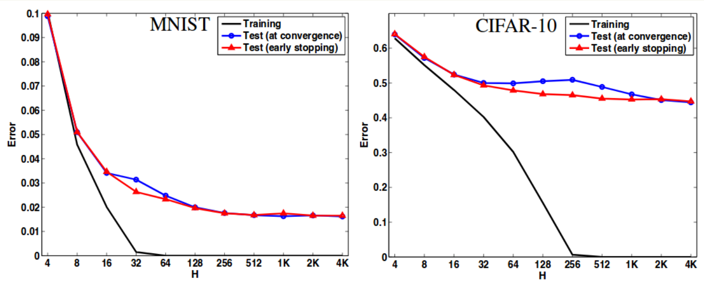
Compromiso Sesgo/Varianza

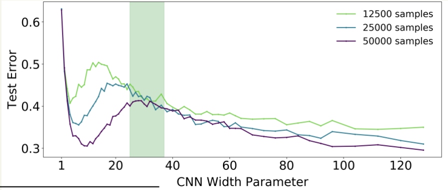
Aprendizaje Clásico



Compromiso Sesgo/Varianza

Deep Learning





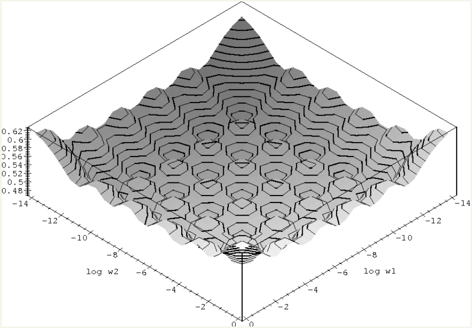
Neyshabur et al. 2017: "Geometry of optimization and implicit regularization in deep learning".

Nakkiran et al. 2019: "Deep Double Descent: Where bigger models and more data hurt".

La sobreparametrización

Otros impactos de la sobreparametrización:

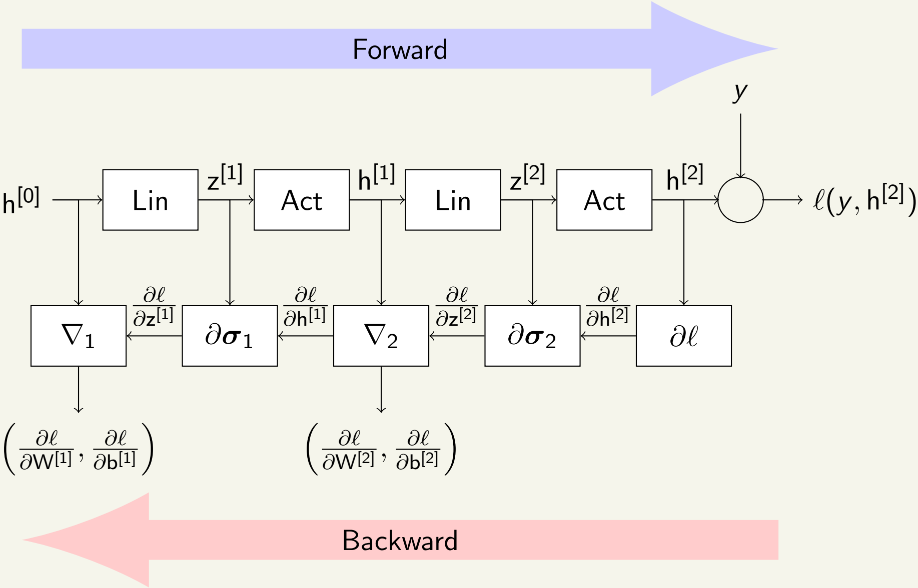
- Muchos *saddle points*.
- La cantidad de mínimos locales es exponencial con la dimensión.
- Incluso los mínimos globales no son todos iguales para la generalización.



Auer et al. 1995: "Exponentially many local minima for single neurons".

Agenda

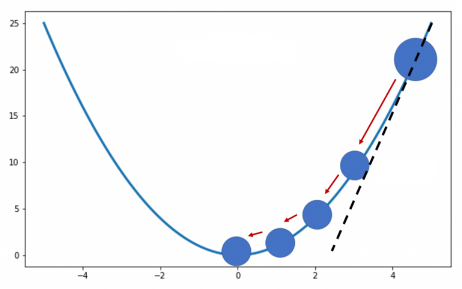
Forward-Backward Propagation



Gradiente Descendente

$$\frac{\partial J(\theta)}{\partial\theta} = \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\frac{\partial\ell(y_i,\mathrm{h}_i^{[L]})}{\partial\theta} = 0$$

$$\theta_{t+1} = \theta_t - \alpha\frac{\partial J(\theta_t)}{\partial\theta}$$



Si tengo muchos datos y muchos parámetros es muy pesado

Gradiente Descendente Estocástico

Pero computar n_{tr} gradientes en cada iteración es pesado, entonces surge la versión estocástica:

$$W_{t+1} = W_t - \alpha \left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t}$$

donde en cada paso se sortea una nueva variable aleatoria (es decir muestras distintas) introduciendo una aleatoriedad al problema. Algunas características son:

- El cómputo es mucho mas liviano que la versión clásica.
- La aleatoriedad me hace converger en una "bola de ruido" (problema de sesgo).
- Mejora la capacidad de generalización agregando ruido al proceso de aprendizaje.

Análisis de convergencia

Supuestos

- Los pares $\{(X_i, Y_i)\}_{i\in\mathbb{N}} \sim (X, Y)$ son i.i.d. e independientes de W_0.
- Los gradientes tienen varianza acotada:

$$\text{var}\left(\frac{\partial L(X, Y; w)}{\partial w}\right) \leq \sigma^2 \qquad \forall\ w \in \Theta$$

- La magnitud $\mathcal{L}(w) = \mathbb{E}[L(X, Y; w)]$ posee un gradiente acotado como $\|\nabla\mathcal{L}(w)\|^2 \leq \mu^2$ para todo $w \in \Theta$.
- $\mathcal{L}(w)$ es fuertemente convexa. Es decir que existe $c > 0$ tal que

$$\mathcal{L}(w_1) \geq \mathcal{L}(w_2) + \langle\nabla\mathcal{L}(w_2); w_1 - w_2\rangle + \frac{c}{2}\|w_1 - w_2\|^2$$

  para todo w_1 y w_2.
- Tecnicismos (existencia de las derivadas, momentos e intercambio).

G. Turinici: "The convergence of the Stochastic Gradient Descent (SGD) : a self-contained proof", 2021.

Análisis de convergencia

Solución óptima

La condición de convexidad asegura que existe un único w^* tal que $\mathcal{L}(w^*) \leq \mathcal{L}(w)$ para todo $w \in \Theta$ y por lo tanto $\nabla\mathcal{L}(w^*) = 0$. La convexidad fuerte asegura entonces que:

$$\mathcal{L}(w) \geq \mathcal{L}(w^*) + \frac{c}{2}\|w - w^*\|^2, \qquad \forall\, w \in \Theta$$

Análisis de convergencia

Solución óptima

La condición de convexidad asegura que existe un único w^* tal que $\mathcal{L}(w^*) \leq \mathcal{L}(w)$ para todo $w \in \Theta$ y por lo tanto $\nabla\mathcal{L}(w^*) = 0$. La convexidad fuerte asegura entonces que:

$$\mathcal{L}(w) \geq \mathcal{L}(w^*) + \frac{c}{2}\|w - w^*\|^2, \qquad \forall\, w \in \Theta$$

$$\begin{aligned}
\mathbb{E}\left[\|W_{t+1} - w^*\|^2\right] &= \mathbb{E}\left[\left\| W_t - w^* - \alpha \left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t} \right\|^2\right] \\
&= \mathbb{E}\left[\|W_t - w^*\|^2\right] + \alpha^2 \mathbb{E}\left[\left\| \left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t} \right\|^2\right] \\
&\quad - 2\alpha \mathbb{E}\left[\left\langle \left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t} ; W_t - w^* \right\rangle\right]
\end{aligned}$$

Análisis de convergencia

Independencia

Como $W_t = f\left(W_0, \{(X_k, Y_k)_{k=1}^{t-1}\}\right)$, es independiente del par (X_t, Y_t).

Análisis de convergencia

Independencia

Como $W_t = f\left(W_0, \{(X_k, Y_k)_{k=1}^{t-1}\}\right)$, es independiente del par (X_t, Y_t).

$$
\begin{aligned}
\mathbb{E}&\left[\left\langle \left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t}; W_t - w^* \right\rangle \middle| W_t = w\right] \\
&= \mathbb{E}\left[\left\langle \frac{\partial L(X_t, Y_t; w)}{\partial w}; w - w^* \right\rangle\right] \\
&= \langle \nabla \mathcal{L}(w); w - w^* \rangle \\
&\geq \mathcal{L}(w) - \mathcal{L}(w^*) + \frac{c}{2}\|w - w^*\|^2 \\
&\geq c\|w - w^*\|^2
\end{aligned}
$$

Análisis de convergencia

$$\mathbb{E}\left[\left\|\left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t}\right\|^2 \middle| W_t = w\right] = \mathbb{E}\left[\left\|\frac{\partial L(X, Y; w)}{\partial w}\right\|^2\right]$$

$$= \text{var}\left(\frac{\partial L(X, Y; w)}{\partial w}\right) + \left\|\mathbb{E}\left[\frac{\partial L(X, Y; w)}{\partial w}\right]\right\|^2 \leq \sigma^2 + \mu^2$$

Análisis de convergencia

$$\mathbb{E}\left[\left\|\left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t}\right\|^2 \middle| W_t = w\right] = \mathbb{E}\left[\left\|\frac{\partial L(X, Y; w)}{\partial w}\right\|^2\right]$$

$$= \operatorname{var}\left(\frac{\partial L(X, Y; w)}{\partial w}\right) + \left\|\mathbb{E}\left[\frac{\partial L(X, Y; w)}{\partial w}\right]\right\|^2 \leq \sigma^2 + \mu^2$$

$$\begin{aligned}
&\mathbb{E}\left[\|W_{t+1} - w^*\|^2\right] \\
&\quad \leq \mathbb{E}\left[\|W_t - w^*\|^2\right] + \alpha^2(\sigma^2 + \mu^2) - 2\alpha c \mathbb{E}\left[\|W_t - w^*\|^2\right] \\
&\quad = (1 - 2\alpha c)\,\mathbb{E}\left[\|W_t - w^*\|^2\right] + \alpha^2(\sigma^2 + \mu^2)
\end{aligned}$$

Análisis de convergencia

$$\mathbb{E}\left[\left\|\left.\frac{\partial L(X_t, Y_t; w)}{\partial w}\right|_{w=W_t}\right\|^2 \middle| W_t = w\right] = \mathbb{E}\left[\left\|\frac{\partial L(X, Y; w)}{\partial w}\right\|^2\right]$$

$$= \operatorname{var}\left(\frac{\partial L(X, Y; w)}{\partial w}\right) + \left\|\mathbb{E}\left[\frac{\partial L(X, Y; w)}{\partial w}\right]\right\|^2 \leq \sigma^2 + \mu^2$$

$$\begin{aligned}
&\mathbb{E}\left[\|W_{t+1} - w^*\|^2\right] \\
&\quad \leq \mathbb{E}\left[\|W_t - w^*\|^2\right] + \alpha^2(\sigma^2 + \mu^2) - 2\alpha c \mathbb{E}\left[\|W_t - w^*\|^2\right] \\
&\quad = (1 - 2\alpha c)\,\mathbb{E}\left[\|W_t - w^*\|^2\right] + \alpha^2(\sigma^2 + \mu^2)
\end{aligned}$$

Bola de ruido

$$\limsup_{t\to\infty} \mathbb{E}\left[\|W_t - w^*\|^2\right] \leq \alpha \cdot \frac{\sigma^2 + \mu^2}{2c}$$

Tradeoff entre bola de ruido y velocidad de convergencia!

Gradiente Descendente Estocástico por minibatch

¿No habrá nada intermedio?

$$W_{t+1} = W_t - \alpha \, \frac{1}{B} \sum_{i=1}^{B} \left. \frac{\partial L\left(X_t^{(i)}, Y_t^{(i)}, w\right)}{\partial w} \right|_{W_t = w}$$

donde en cada paso se sortean B una nuevas variables aleatorias i.i.d. El "batchsize" B nos permite explotar el tradeoff según nuestra conveniencia. En el análisis de la bola de ruido, la función $\mathcal{L}(w)$ es la misma, lo único que cambia es σ^2:

$$\operatorname{var}\left(\frac{\partial \frac{1}{B} \sum_{i=1}^{B} L\left(X^{(i)}, Y^{(i)}; w\right)}{\partial w} \right) \leq \frac{\sigma^2}{B}$$

Obteniendo una bola de ruido de la forma:

$$\limsup_{t \to \infty} \mathbb{E}\left[\|W_t - w^*\|^2\right] \leq \alpha \cdot \frac{\frac{\sigma^2}{B} + \mu^2}{2c}$$

Sobre el batchsize

- Batchs grandes proporcionan una estimación más precisa del gradiente, pero de forma mas pesada a nivel computacional.
- Las arquitecturas multicore poseen un mínimo valor del batchsize debajo del cual no hay reducción en el tiempo de procesamiento.
- Si todos las muestras del batch se procesan en paralelo, entonces la cantidad de memoria escala con el tamaño del batch, limitando superiormente el batchsize.
- Algunos tipos de hardware logran un mejor tiempo de ejecución con tamaños específicos de arrays. Especialmente las GPU suelen ser más rápidas con batchsize que sean potencias de 2.
- Los batchs pequeños ofrecen un efecto de regularización debido al ruido que se agrega al proceso de aprendizaje. La generalización suele ser mejor para un batchsize de 1. Pero batchs tan pequeños requieren bajos learning rates lo que puede demorar el proceso de aprendizaje.

Goodfellow, Bengio and Courville: "Deep Learning", chapter 8.

Sobre el batchsize

En la práctica...

Se utilizan datos distintos hasta que se acaban, realizando en principio n_{tr}/B pasos del algoritmo. Como esto suele ser insuficiente, los datos se mezclan y se vuelven a usar (construyendo batches distintos). A cada pasada de todos los datos se lo llama epoch.

Hiperparámetros de esta etapa:

- Batch size (B)
- Número de epochs (o condición de stop)

Diferentes tipos de optimizadores

Variantes del gradiente descendente estocástico (SGD) implementadas en Keras:

- SGD with momentum
- Nesterov
- RMSProp
- Adagrad
- Adadelta
- Adam
- Adamax
- AMSGrad
- Nadam

Momentum: Exponentially Weighted Averages

Objetivo

Bajar la bola de ruido con alta velocidad de aprendizaje (α grande) sin aumentar la carga computacional (batchsize moderado). Es decir, buscar reemplazos para $\mathrm{g}_t = \frac{1}{B}\sum_{i=1}^{B} \frac{\partial L_i(\theta_{t-1})}{\partial \theta}$.

Momentum (no le creas tanto): $\mathrm{m}_t = \beta_1 \mathrm{m}_{t-1} + (1-\beta_1)\mathrm{g}_t$ con $\beta_1 \in [0, 1)$ (típico $\beta_1 = 0.9$).

Momentum: Exponentially Weighted Averages

Objetivo

Bajar la bola de ruido con alta velocidad de aprendizaje (α grande) sin aumentar la carga computacional (batchsize moderado). Es decir, buscar reemplazos para $\mathrm{g}_t = \frac{1}{B}\sum_{i=1}^{B} \frac{\partial L_i(\theta_{t-1})}{\partial \theta}$.

Momentum (no le creas tanto): $\mathrm{m}_t = \beta_1 \mathrm{m}_{t-1} + (1-\beta_1)\mathrm{g}_t$ con $\beta_1 \in [0, 1)$ (típico $\beta_1 = 0.9$).

$$\mathrm{m}_5 = (1-\beta_1)\mathrm{g}_5 + (1-\beta_1)\beta_1\mathrm{g}_4 + (1-\beta_1)\beta_1^2\mathrm{g}_3 + (1-\beta_1)\beta_1^3\mathrm{g}_2 + (1-\beta_1)\beta_1^4\mathrm{g}_1$$

Momentum: Exponentially Weighted Averages

Objetivo

Bajar la bola de ruido con alta velocidad de aprendizaje (α grande) sin aumentar la carga computacional (batchsize moderado). Es decir, buscar reemplazos para $\mathrm{g}_t = \frac{1}{B}\sum_{i=1}^{B} \frac{\partial L_i(\theta_{t-1})}{\partial \theta}$.

Momentum (no le creas tanto): $\mathrm{m}_t = \beta_1 \mathrm{m}_{t-1} + (1-\beta_1)\mathrm{g}_t$ con $\beta_1 \in [0,1)$ (típico $\beta_1 = 0.9$).

$$\mathrm{m}_5 = (1-\beta_1)\mathrm{g}_5 + (1-\beta_1)\beta_1 \mathrm{g}_4 + (1-\beta_1)\beta_1^2 \mathrm{g}_3 + (1-\beta_1)\beta_1^3 \mathrm{g}_2 + (1-\beta_1)\beta_1^4 \mathrm{g}_1$$

$$\mathrm{m}_t = (1-\beta_1)\sum_{k=0}^{t-1} \beta_1^k \mathrm{g}_{t-k}$$

Respuesta: $\theta_t = \theta_{t-1} - \alpha \mathrm{m}_t$.
Learning rate efectivo: Fue atenuado $\alpha_{\text{ef}} = \alpha(1-\beta_1)$.

$$\theta_t = \theta_{t-1} - \alpha\beta_1 \mathrm{m}_{t-1} - \alpha(1-\beta_1)\mathrm{g}_t$$

Bias Correction

Momentum subestima mucho los gradientes en las primeras iteraciones:

$$\begin{aligned}
m_1 &= (1-\beta_1)g_1 = 0.1g_1 \\
m_2 &= (1-\beta_1)g_2 + (1-\beta_1)\beta_1 g_1 = 0.1g_2 + 0.09g_1
\end{aligned}$$

Bias Correction

Momentum subestima mucho los gradientes en las primeras iteraciones:

$$\begin{aligned} \mathrm{m}_1 &= (1-\beta_1)\mathrm{g}_1 = 0.1\mathrm{g}_1 \\ \mathrm{m}_2 &= (1-\beta_1)\mathrm{g}_2 + (1-\beta_1)\beta_1\mathrm{g}_1 = 0.1\mathrm{g}_2 + 0.09\mathrm{g}_1 \end{aligned}$$

Solución: Verifiquemos que sea una combinación convexa de gradientes.

$$\hat{\mathrm{m}}_t = \frac{(1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k \mathrm{g}_{t-k}}{(1-\beta_1)\sum_{k=0}^{t-1}\beta_1^k} = \frac{\mathrm{m}_t}{1-\beta_1^t}$$

Respuesta: $\theta_t = \theta_{t-1} - \alpha\hat{\mathrm{m}}_t$.
Learning rate efectivo: Atenuación paulatina $\alpha_{\mathrm{ef}} = \alpha\frac{1-\beta_1}{1-\beta_1^t}$.

Nesterov's accelerated gradient

Classical momentum: $\theta_t = \theta_{t-1} - \alpha\beta_1 \mathsf{m}_{t-1} - \alpha(1-\beta_1)\frac{\partial L(\theta_{t-1})}{\partial \theta}$

Sutskever et al. 2013: "On the importance of initialization and momentum in deep learning".

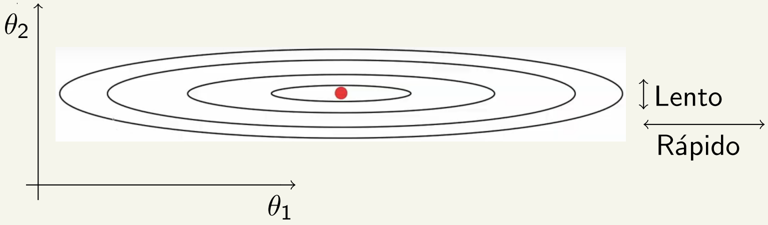
Nesterov's accelerated gradient

Classical momentum: $\theta_t = \theta_{t-1} - \alpha\beta_1 \mathrm{m}_{t-1} - \alpha(1-\beta_1)\frac{\partial L(\theta_{t-1})}{\partial \theta}$
Nesterov momentum: $\theta_t = \theta_{t-1} - \alpha\beta_1 \mathrm{m}_{t-1} - \alpha(1-\beta_1)\frac{\partial L(\theta_{t-1}-\alpha\beta_1 \mathrm{m}_{t-1})}{\partial \theta}$

La idea del gradiente es que me diga cuanto y hacia donde me tengo que mover para acercarme al mínimo. Nesterov usa toda la información que tengo actualmente antes de elegir cuanto y hacia donde moverse.

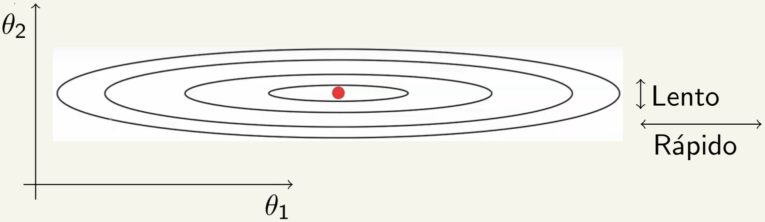
Sutskever et al. 2013: "On the importance of initialization and momentum in deep learning".

Adaptive Gradient (Adagrad)



Duchi et al. 2011: "Adaptive subgradient methods for online learning and stochastic optimization".

Adaptive Gradient (Adagrad)



Solución: Usar la historia del gradiente para escalarlo. Se define un parámetro que evite errores numéricos $\epsilon \sim 10^{-8}$.

Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{\mathrm{g}_t}{\sqrt{\sum_{\tau=1}^{t} \mathrm{g}_\tau^2} + \epsilon}$.

Learning rate efectivo: Se adapta a cada componente.

Duchi et al. 2011: "Adaptive subgradient methods for online learning and stochastic optimization".

Root Mean Square Propagation

RMSProp

Exponentially weighted averages para los cuadrados: $\mathrm{v}_t = \beta_2 \mathrm{v}_{t-1} + (1-\beta_2)\mathrm{g}_t^2$.
Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{\mathrm{g}_t}{\sqrt{\mathrm{v}_t + \epsilon}}$.

RMSProp with bias correction

Misma idea: $\hat{\mathrm{v}}_t = \frac{\mathrm{v}_t}{1-\beta_2^t}$.
Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{\mathrm{g}_t}{\sqrt{\hat{\mathrm{v}}_t + \epsilon}}$.

ADAM

Combinar RMSProp con momentum (ambos con bias correction).
Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{\mathrm{m}_t}{\sqrt{\hat{\mathrm{v}}_t + \epsilon}}$.

Kingma-Lei Ba 2015: "ADAM: A method for stochastic optimization".

Variantes de ADAM

ADAMAX

Misma idea de RMSProp pero con norma infinito: $u_t = \max\{\beta_2 u_{t-1}, |g_t|\}$.
Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{m_t}{u_t}$.

AMSGrad

Variante sobre los v_t que corrige algunos problemas de convergencia propios de RMSProp: $\tilde{v}_t = \max\{\tilde{v}_{t-1}; v_t\}$.
Respuesta: $\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{\tilde{v}_t} + \epsilon}$.

NADAM

Si ADAM es RMSProp con momentum, NADAM es RMSProp con Nesterov momentum.

Kingma-Lei Ba 2015: "ADAM: A method for stochastic optimization".
Reddi et al. 2018: "On the convergence of ADAM and beyond".
Dozat 2016: "Incorporating Nesterov Momentum into ADAM".

Adadelta

Deja que el learning rate se elija solo!

$$
\begin{aligned}
\mathsf{s}_t &= \beta_2 \mathsf{s}_{t-1} + (1-\beta_2)\mathsf{g}_t^2 \\
\varDelta_t &= \beta_2 \varDelta_{t-1} + (1-\beta_2)\theta_t^2 \\
\theta_t &= \theta_{t-1} - \alpha_0 \sqrt{\frac{\varDelta_{t-1}+\epsilon}{\mathsf{s}_t+\epsilon}}\mathsf{g}_t
\end{aligned}
$$

Remark

α_0 NO ES el learning-rate! Se suele setear en $\alpha_0 = 1$ (mientras que los α suelen ser bastante más chicos). De hecho tanto en el artículo original como en algunas implementaciones no está este hiperparámetro.

Zeiler 2012: "ADADELTA: An adaptive learning rate method".

Learning rate decay

Puedo ir reduciendo la bola de ruido a medida que voy aprendiendo. Sea $e_t = 1 + \lfloor \frac{tB}{n_{\text{tr}}} \rfloor$ el número de epoch que estoy en la iteración t, luego algunas de las expresiones más utilizadas son:

- $\alpha = \alpha_0 \frac{1+\delta}{1+\delta e_t}$
- $\alpha = \alpha_0 \gamma^{e_t - 1}$
- $\alpha = \frac{\alpha_0}{\sqrt{e_t}}$
- Definir α constante por intervalos de e_t (de forma decreciente)