DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Entrenamiento de redes neuronales

Fernando Berzal, berzal@acm.org

Entrenamiento de redes neuronales

- Modos de entrenamiento
- Preprocesamiento de los datos
- Funciones de activación
- Inicialización de los pesos
- Normalización por lotes [batch normalization]
- Tasas de aprendizaje

En la práctica…

Algoritmo de aprendizaje de redes multicapa

Aspectos que debemos considerar en su diseño:

- **Parámetros:** ¿Qué topología de red utilizamos?...
- **Optimización:** ¿Cómo obtenemos los pesos?
- **Generalización:** ¿Cómo conseguimos que la red funcione bien con datos distintos a los del conjunto de entrenamiento?
- **Invarianza:** ¿Cómo conseguimos que la red sea robusta frente a transformaciones comunes en los datos?

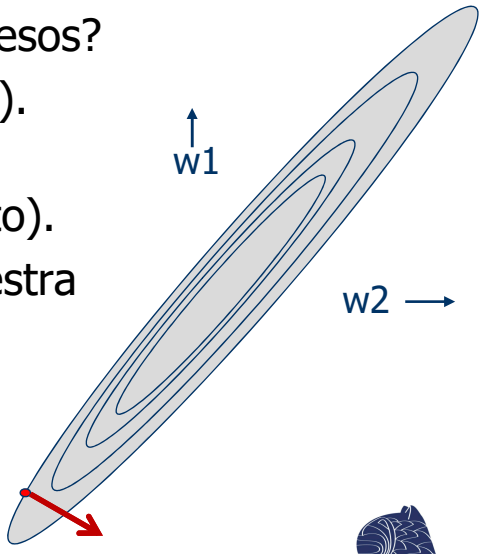
DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Entrenamiento de redes neuronales
Modos de entrenamiento

Modos de entrenamiento

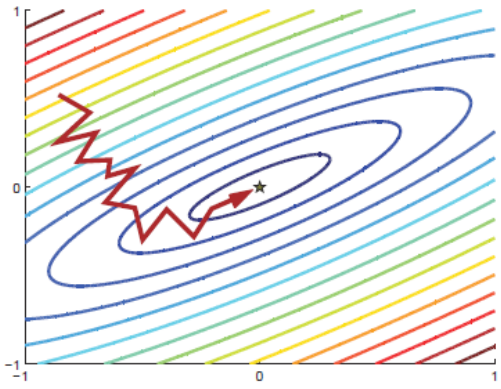
¿Con qué frecuencia se ajustan los pesos?

- **Online** (después de cada ejemplo).
- **Batch** (después de cada recorrido sobre el conjunto de entrenamiento).
- **Mini-batch** (después de una muestra del conjunto de entrenamiento)

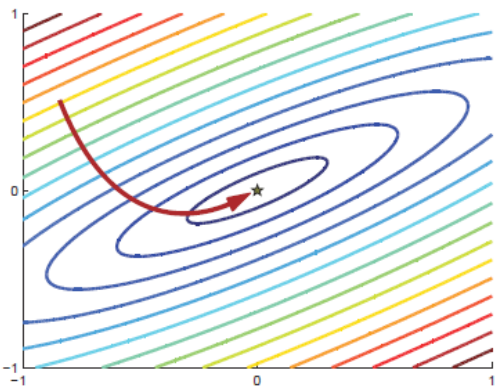


Modos de entrenamiento

Online learning vs. Batch learning



Aprendizaje online



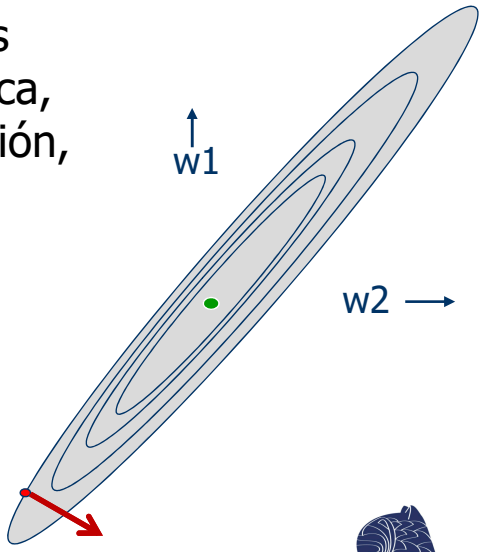
Aprendizaje por lotes

Modos de entrenamiento

"Batch learning", a.k.a. Full-batch gradient descent

Aunque para las neuronas no lineales la superficie de error no sea cuadrática, localmente nos vale como aproximación, por lo que el aprendizaje "full batch" sigue presentando problemas:

La dirección de máxima pendiente del gradiente no apunta al mínimo salvo que la elipse sea un círculo.

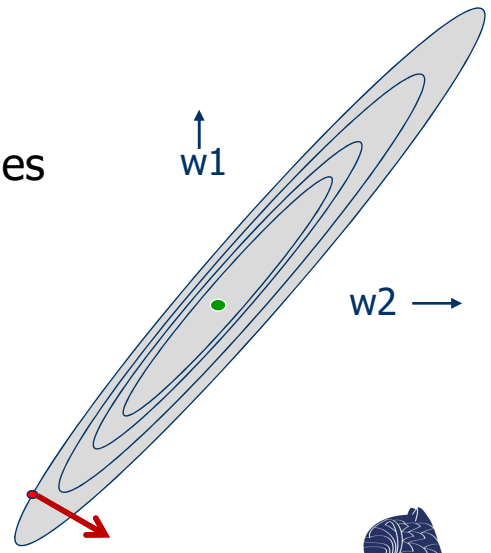


Modos de entrenamiento

"Batch learning", a.k.a. Full-batch gradient descent

Lo que nos gustaría conseguir:

- Movernos más rápido en direcciones con gradientes pequeños pero consistentes.
- Movernos más despacio en direcciones con gradientes grandes pero inconsistentes.



Modos de entrenamiento

Online learning

Estimación del gradiente a partir del error observado para cada ejemplo de entrenamiento.

Ventajas del aprendizaje online:

- Mucho más rápido que el aprendizaje por lotes.
- Suele obtener mejores soluciones.
- Facilita adaptarse a cambios.

Modos de entrenamiento

Online learning vs. Batch learning

Pese a las ventajas del aprendizaje "online", existen motivos que justifican utilizar el aprendizaje por lotes.

Ventajas del aprendizaje por lotes:

- Condiciones de convergencia bien conocidas.
- Muchas técnicas de optimización sólo funcionan con batch learning (p.ej. gradientes conjugados).
- Análisis teórico (dinámica y convergencia).

Modos de entrenamiento

Mini-batch gradient descent

Si existe redundancia en el conjunto de entrenamiento, el gradiente de su primera mitad será similar al de su segunda mitad, por lo que podemos actualizar los pesos usando una mitad y volver a obtener un gradiente para esos pesos actualizados usando la segunda mitad.

La versión extrema de esta estrategia es el "online learning" (actualizar los pesos después de cada ejemplo).

Ambas son formas de **gradiente descendente estocástico** [stochastic gradient descent].

Modos de entrenamiento

Mini-batch gradient descent

El uso de mini-lotes (balanceados para las distintas clases en problemas de aprendizaje supervisado) suele ser mejor que el aprendizaje "online".

- Se requieren menos cálculos para actualizar los pesos.
- El cálculo simultáneo del gradiente para muchos casos puede realizarse utilizando operaciones con matrices que se pueden implementar de forma muy eficiente utilizando GPUs.

Modos de entrenamiento

Efficient BackProp [LeCun et al.]

Consejos para la implementación de backpropagation

RECOMENDACIÓN: ENTRENAMIENTO

Elegir los ejemplos que proporcionan mayor información para el entrenamiento de la red.

- Barajar los ejemplos para que ejemplos consecutivos (casi) nunca pertenezcan a la misma clase.
- Presentar los ejemplos que producen mayores errores con mayor frecuencia que los ejemplos que producen menos errores

DECSAI

Departamento de Ciencias de la Computación e I.A.

Universidad de Granada

Entrenamiento de redes neuronales

Preprocesamiento de los datos

Preprocesamiento de los datos

Efficient BackProp [LeCun et al.]

Consejos para la implementación de backpropagation

RECOMENDACIÓN: PREPROCESAMIENTO

Normalización de las entradas (p.ej. z-scores).

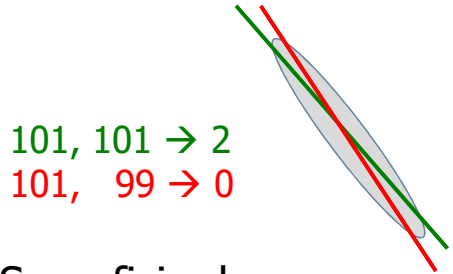
- La media de cada variable de entrada en el conjunto de entrenamiento debería ser cercana a cero.
- La escala de las variables de entrada debería ajustarse para que sus covarianzas sean similares.
- Si es posible, las variables de entrada deberían estar decorreladas (sin correlación).

Preprocesamiento de los datos

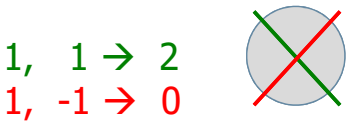
Efficient BackProp

Consejos para la implementación de backpropagation

EJEMPLO: Desplazamiento (media 0)



Superficie de error
(datos originales)



Superficie de error
(datos con media 0)

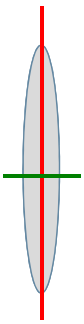
Preprocesamiento de los datos

Efficient BackProp

Consejos para la implementación de backpropagation

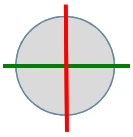
EJEMPLO: Escalado

0.1, 10 → 2
0.1, -10 → 0



Superficie de error
(datos originales)

1, 1 → 2
1, -1 → 0



Superficie de error
(datos escalados)

Preprocesamiento de los datos

Efficient BackProp

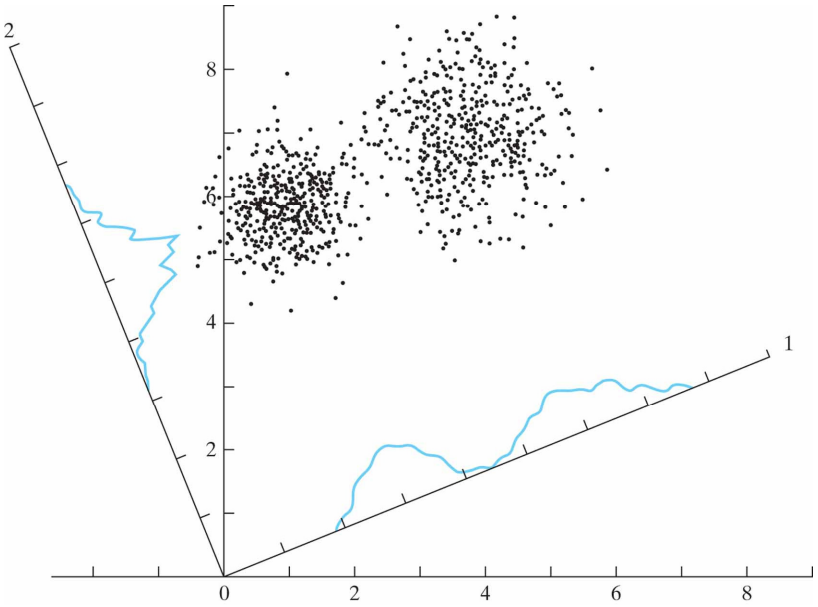
Consejos para la implementación de backpropagation

Un método para decorrelar las variables de entrada:
Análisis de componentes principales [PCA]

- Permite reducir la dimensionalidad de los datos (eliminando componentes con menores eigenvalues).
- Permite convertir una superficie de error elíptica en una circular, en la que el gradiente apunta directamente al mínimo (dividiendo los componentes principales por las raíces cuadradas de sus respectivos eigenvalues).

Preprocesamiento de los datos

Análisis de componentes principales [PCA]

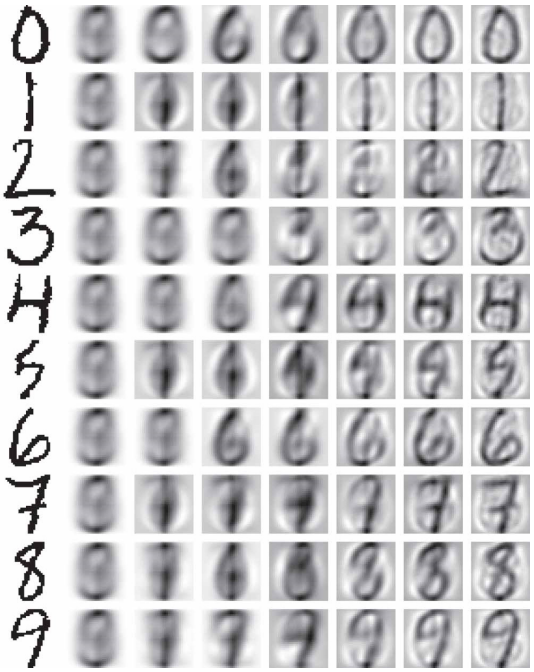


[Haykin: "Neural Networks and Learning Machines", 3rd edition]

Preprocesamiento de los datos

Análisis de componentes principales [PCA]

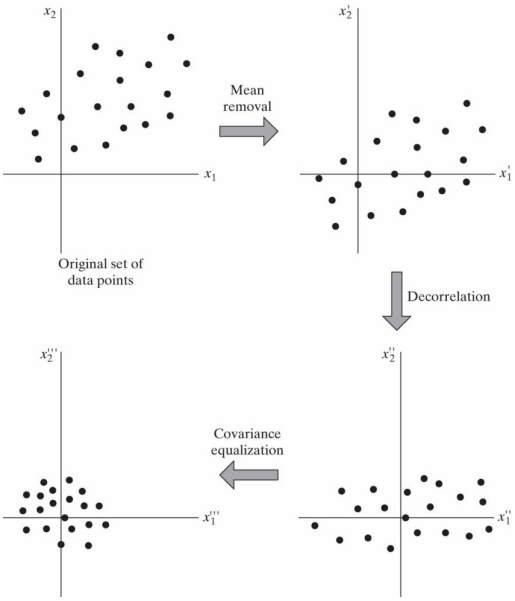
Compresión de imágenes usando componentes principales



[Haykin: "Neural Networks and Learning Machines", 3rd edition]

Preprocesamiento de los datos

Análisis de componentes principales [PCA]

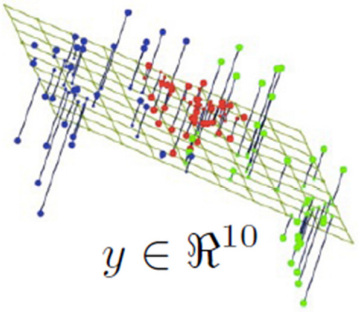


[Haykin: "Neural Networks and Learning Machines", 3rd edition]

Apéndice
Reducción de dimensionalidad

Proyección en un espacio de menos dimensiones

$x \in \Re^{64 \times 64} = \Re^{4096}$



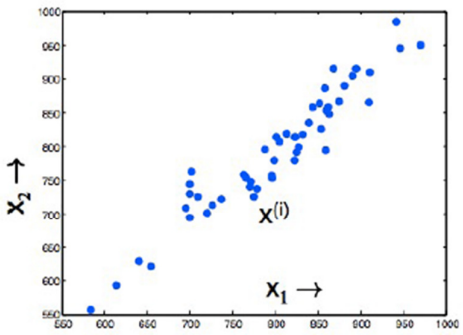
$y \in \Re^{10}$

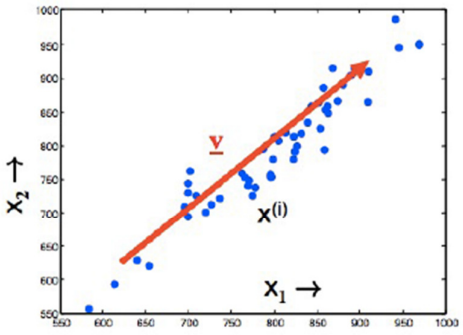
$$y = Ux$$

http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Análisis de componentes principales





$$\vec{x} = [x_1, x_2]$$

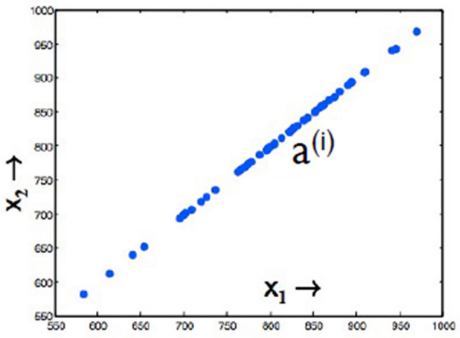
$$\vec{x} \approx s\vec{v} = s[v_1, v_2]$$

http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Análisis de componentes principales

Se escogen los vectores que minimizan la varianza de los residuos



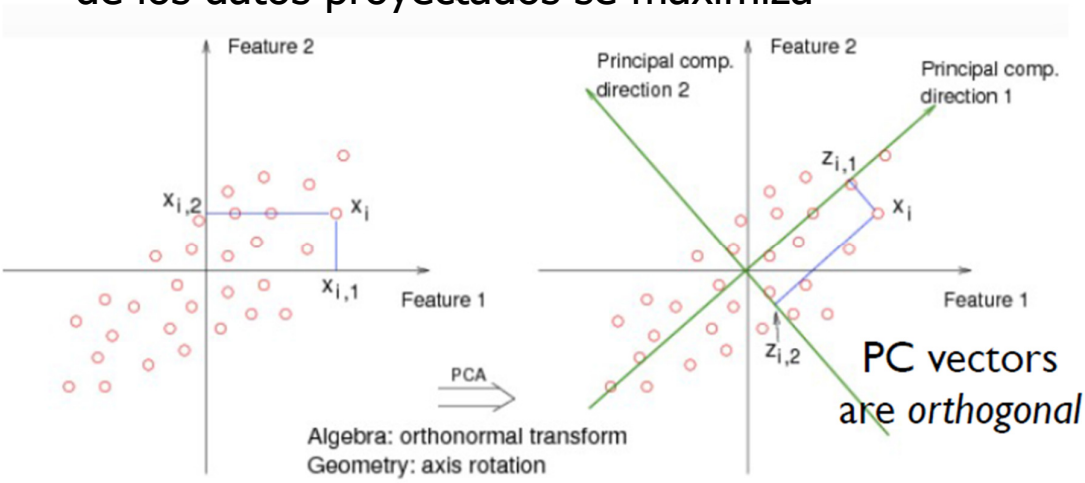
$$\min_{a,v} \sum_i (x^{(i)} - a^{(i)} v)^2$$

http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Análisis de componentes principales

Se proyecta en subespacios en los que la varianza de los datos proyectados se maximiza



http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Análisis de componentes principales

Cálculo de los componentes principales:

1. Se calcula la matriz de covarianza Σ de los datos.
2. Se calculan los k mayores eigenvectores de la matriz de covarianza Σ (los componentes principales).

Pero este cálculo puede ser demasiado costoso cuando se hace iterativamente sobre la matriz de covarianza…

… por lo que se suele utilizar la descomposición de valores singulares [SVD] para calcular los componentes principales.

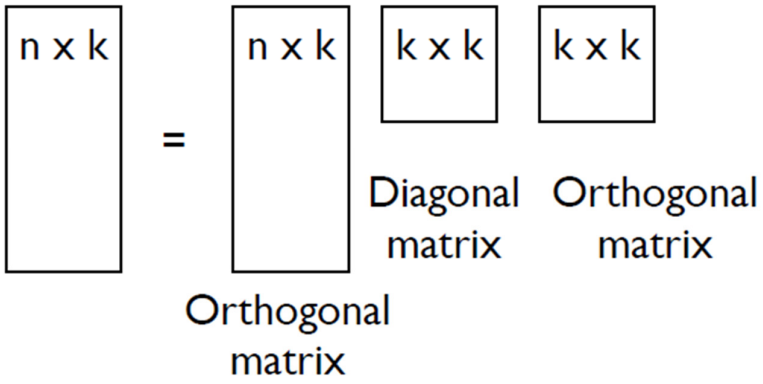
http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Descomposición de valores singulares [SVD]

Los eigenvectores son las columnas de la matriz U.

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$

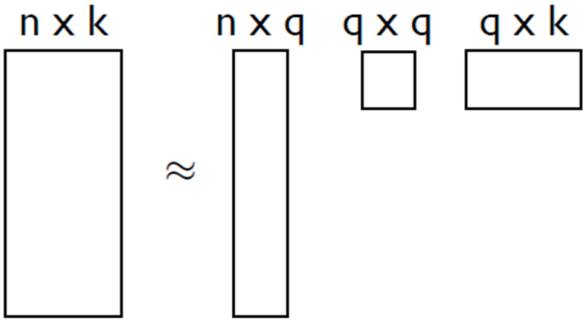


http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Se reduce la dimensionalidad quedándonos sólo con los q primeros componentes principales (q<k):

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{D}}\tilde{\mathbf{V}}^{\mathsf{T}}$$



http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

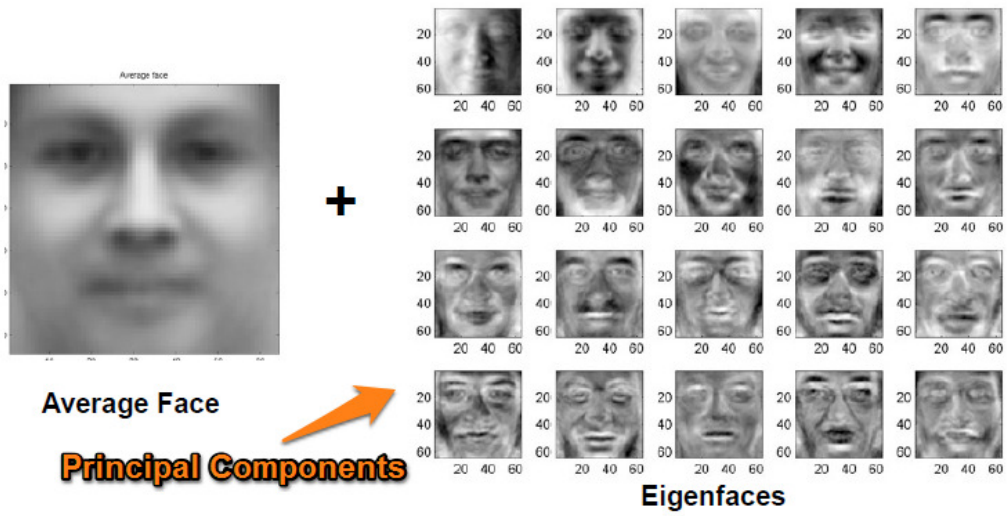
Ejemplo: Reconocimiento de caras

64x64 images of faces = 4096 dimensional data

http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

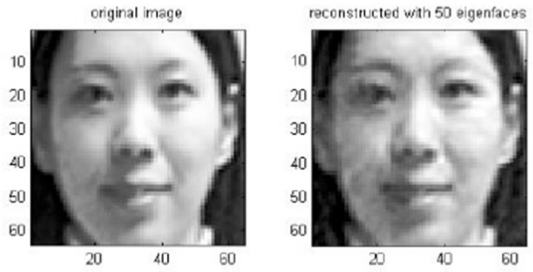
Ejemplo: Reconocimiento de caras



http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

Apéndice
Reducción de dimensionalidad

Ejemplo: Reconocimiento de caras



Los 50 primeros eigenvectores describen el 90% de la varianza, por lo que podemos quedarnos sólo con esas 50 dimensiones para reconstruir los datos sin perder demasiada calidad (de 4096 a 50 dimensiones ;-).

http://www.bigdataexaminer.com/understanding-dimensionality-reduction-principal-component-analysis-and-singular-value-decomposition/

DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Entrenamiento de redes neuronales
Funciones de activación

Funciones de activación

Funciones de activación

RECOMENDACIONES [LeCun et al.]

- Las sigmoides simétricas, como la tangente hiperbólica, suelen converger más rápidamente que la función logística: **tanh(x)=2*logistic(2x)-1**.
- Función recomendada: f(x)= 1.7159 tanh(2x/3).
- En ocasiones, resulta útil añadir un pequeño término lineal para evitar zonas planas (de gradiente 0), p.ej. f(x)= tanh(x) + ax

Funciones de activación

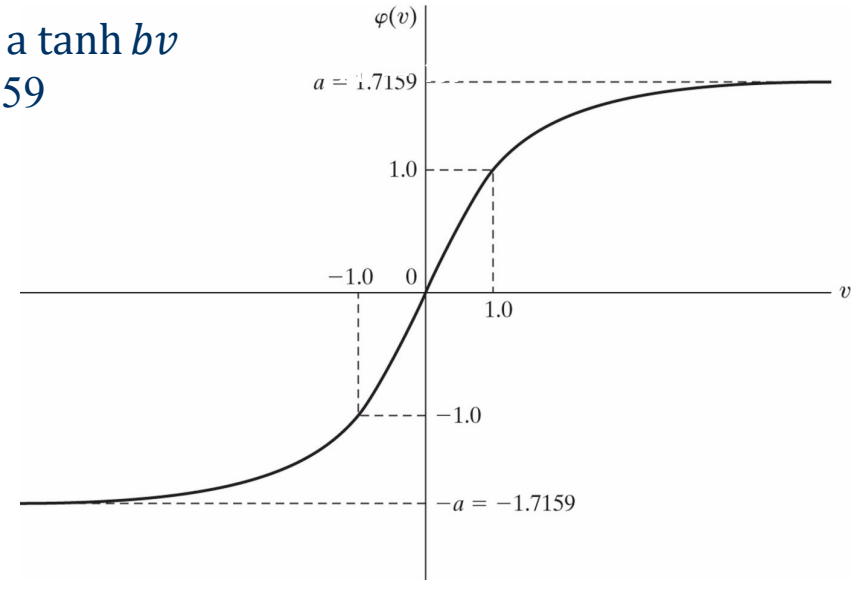
Funciones de activación

Función de activación recomendada [LeCun et al.]

$\varphi(v) = a \tanh bv$

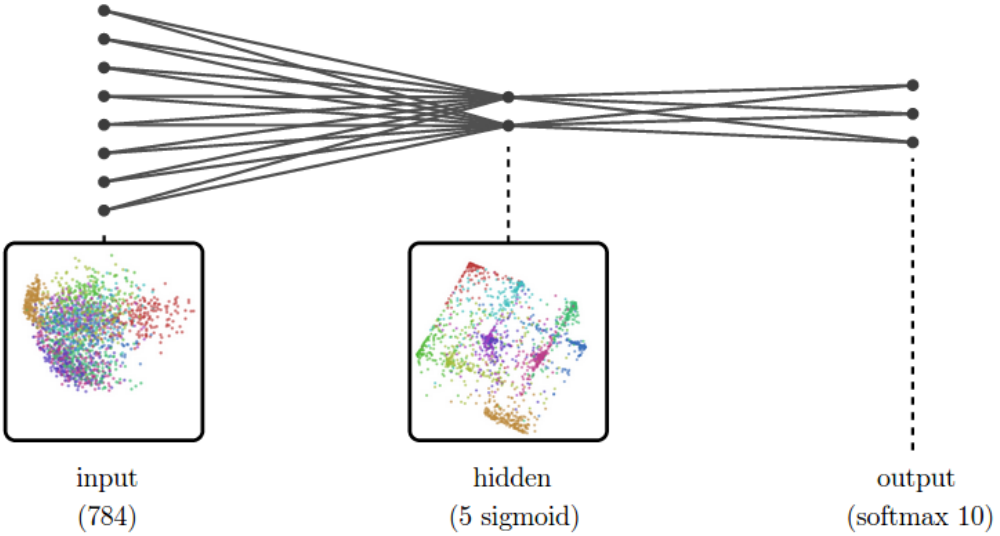
$a = 1.7159$

$b = 2/3$



Funciones de activación

Funciones de activación sigmoidales



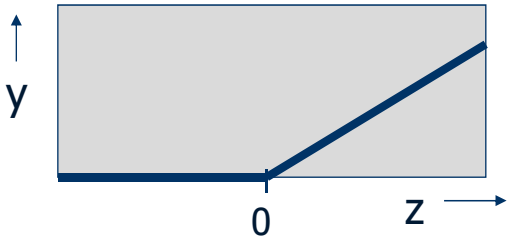
Funciones de activación

Funciones de activación: ReLU

En ocasiones, especialmente en "deep learning",
se utilizan unidades lineales rectificadas (ReLU)
porque su entrenamiento suele ser mucho más rápido:

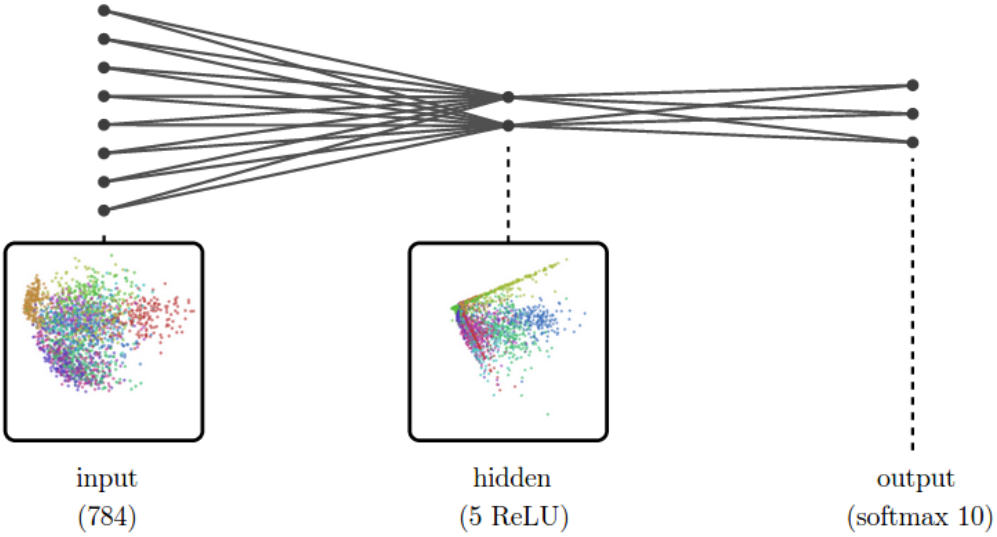
$$z = \sum_i x_i w_i$$

$$y = \begin{cases} z & si\ z \geq 0 \\ 0 & en\ otro\ caso \end{cases}$$



Funciones de activación

Funciones de activación: ReLU



Funciones de activación

Funciones de activación: ReLU

Problema: Dying RELUs

La mitad de las neuronas dejan de hacer nada (salida fija a 0), especialmente si se utiliza una tasa de aprendizaje elevada…

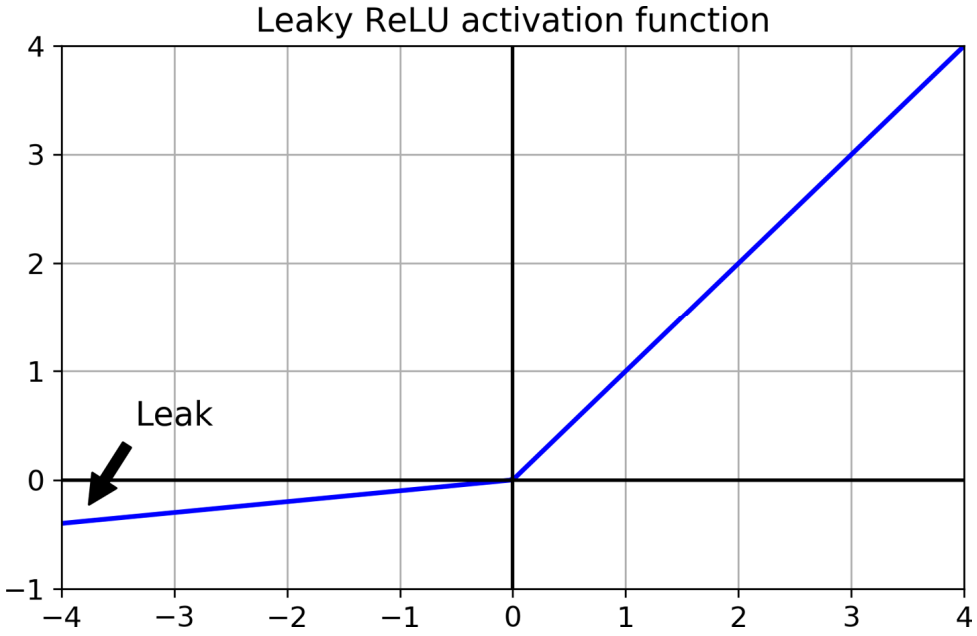
Solución: Leaky RELU

$\max\{0,z\} \rightarrow \max\{\alpha z, z\}$, p.ej. $\alpha \in [0.01, 0.3]$

Las "leaky RELU" nunca mueren, aunque pueden entrar en coma del que tal vez despierten…

Funciones de activación

Funciones de activación: Leaky ReLU



Funciones de activación

Funciones de activación: Leaky ReLU

Variantes @ arXiv'2015

- **Randomized leaky ReLU [RReLU]:**
  α se elige aleatoriamente durante el entrenamiento.

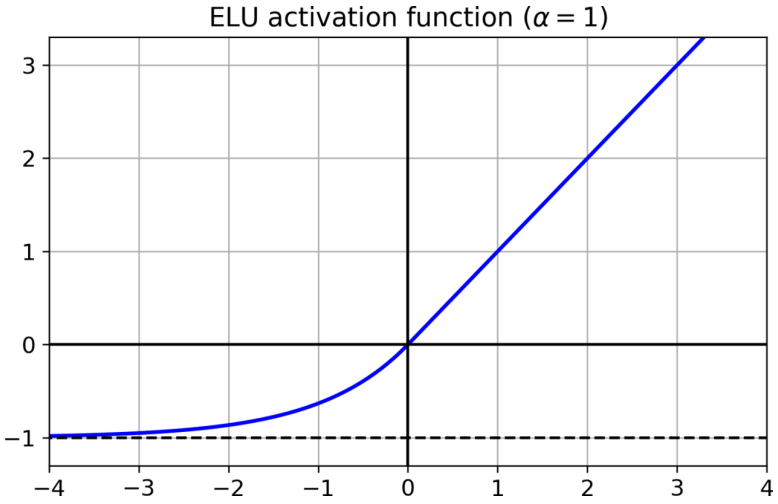
- **Parametric leaky ReLU [PReLU]:**
  α se entrena como un parámetro más de la red.

Funciones de activación

Funciones de activación: ELU

[Exponential Linear Unit] @ ICLR'2016

$$\mathrm{ELU}_{\alpha}(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$



Funciones de activación

Funciones de activación: SELU

[Scaled Exponential Linear Unit] @ NIPS'2017

Variante de ELU en la que una red multicapa de tipo feed-forward (sin realimentación) ni "skip connections" se autonormaliza: la salida cada capa tiende a preservar media 0 y desviación 1 durante el entrenamiento, lo que previene problemas con el gradiente [vanishing/exploding gradients].

Funciones de activación

Funciones de activación

¿Cuál elegir?

SELU > ELU > leaky ReLU > ReLU > tanh > logistic

NB:
Muchas bibliotecas están optimizadas para utilizar ReLU.

Salida de la red

Efficient BackProp [LeCun et al.]
Consejos para la implementación de backpropagation

RECOMENDACIÓN: VALORES DE SALIDA

- En problemas de clasificación, {+1, -1}.
- Para evitar problemas de saturación en las unidades de salida, se pueden elegir puntos en el rango de la sigmoide que maximicen su segunda derivada.

NOTA:
En problemas de regresión, salidas no acotadas, p.ej. neuronas lineales.

Entrenamiento de redes neuronales
Inicialización de los pesos

Inicialización de los pesos

Inicialización aleatoria de los pesos

- Si dos neuronas tienen exactamente los mismos pesos, siempre tendrán el mismo gradiente, por lo que no serán capaces de aprender características diferentes.
- Se rompe la simetría inicializando los pesos con valores aleatorios (pequeños).

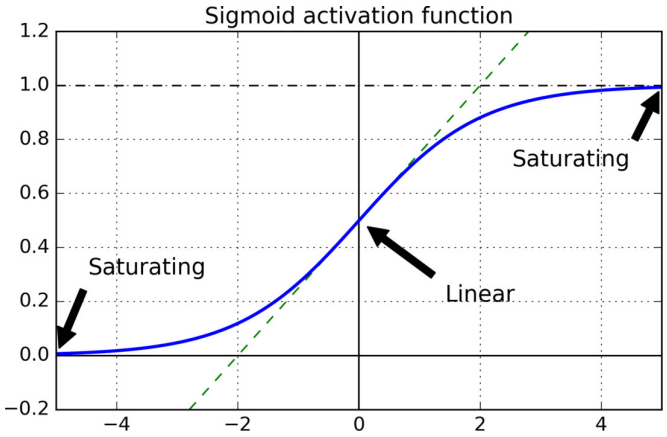
Inicialización de los pesos

Enfoque tradicional (siglo XX)

- Distribución normal N(0,1) con media 0 y desviación 1.
- **Problema:** Esta inicialización, combinada con la función de activación logística, ocasiona que la varianza de las salidas en cada capa sea mayor que la varianza de sus entradas.
- **Consecuencia:** La varianza aumenta hasta que se saturan las neuronas de las capas finales, lo que dificulta el aprendizaje…

Inicialización de los pesos

Enfoque tradicional (siglo XX)



Cuando la derivada de la función de activación es cercana a 0, nos quedamos sin gradiente del error que propagar hacia atrás :-(

Inicialización de los pesos

Enfoque moderno (siglo XXI)

- Si una neurona tiene muchas conexiones de entrada ("fan-in" elevado), pequeños cambios en muchos de sus pesos de entrada pueden hacer que nos pasemos.
- Normalmente, queremos pesos más pequeños cuando el "fan-in" es alto, por lo que se suelen inicializar los pesos aleatorios proporcionalmente a 1/sqrt(fan-in).

NOTA:
La misma regla puede aplicarse a la tasa de aprendizaje.

Inicialización de los pesos

Enfoque moderno (siglo XXI)

Consejos para la implementación de backpropagation

RECOMENDACIÓN [LeCun et al.], desde los años 90 !!!

Asumiendo que el conjunto de entrenamiento se ha normalizado y que se usa la función de activación $\varphi(v) = 1.7151 \tanh(2v/3)$, los pesos se deberían inicializar aleatoriamente utilizando una distribución (p.ej. uniforme) con media **0** y desviación estándar $\sigma_w = 1/\mathrm{sqrt}(m)$, donde m es el fan-in del nodo (número de conexiones que llegan al nodo).

Inicialización de los pesos

Enfoque moderno (siglo XXI)

Inicialización de Xavier

RECOMENDACIÓN [Glorot & Bengio, 2010]:

Para que las señales se propaguen correctamente, necesitamos que los gradientes tengan la misma varianza antes y después de cada capa, algo que no se puede garantizar si no tienen el mismo número de entradas que de salidas (fan-in distinto a fan-out).

Compromiso: Distribución normal con media **0** y desviación estándar σ_w**=1/sqrt(m)**, donde m es la media del fan-in y del fan-out.

Alternativa (p.ej. Keras):

Distribución uniforme entre **-r** y **+r**, con **r=sqrt(3/m)**

DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Entrenamiento de redes neuronales

Normalización por lotes [batch normalization]

Normalización por lotes

Técnica propuesta por Sergey Ioffe y Christian Szegedy en 2015 para evitar problemas relacionados con el gradiente del error en redes multicapa [vanishing/exploding gradients]:

IDEA:
Centrar y normalizar la entrada neta de cada capa oculta

MECANISMO:

Dos vectores de parámetros adicionales por capa (escala y desplazamiento), que se entrenan con backpropagation.

Normalización por lotes

Se estima la media y desviación estándar de cada minilote B durante el entrenamiento de la red:

$$\boldsymbol{\mu}_B = \frac{1}{m_B} \sum_{i=1}^{m_B} \mathbf{x}^{(i)}$$

$$\boldsymbol{\sigma}_B^{\,2} = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_B\right)^2$$

Se estandariza la entrada y se reescala (escala γ, desplazamiento β):

$$\widehat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}_B}{\sqrt{\boldsymbol{\sigma}_B^{\,2} + \varepsilon}} \qquad \mathbf{z}^{(i)} = \boldsymbol{\gamma} \otimes \widehat{\mathbf{x}}^{(i)} + \boldsymbol{\beta}$$

Normalización por lotes

Durante el entrenamiento, la muestra correspondiente a cada minilote sirve para estimar medias y desviaciones.

¿Qué sucede luego?

Podríamos utilizar el conjunto de entrenamiento completo para calcular los estadísticos de entrada de cada capa (media μ y desviación σ).

Normalmente, en la práctica, se estiman utilizando medias móviles de las estimaciones para cada minilote.

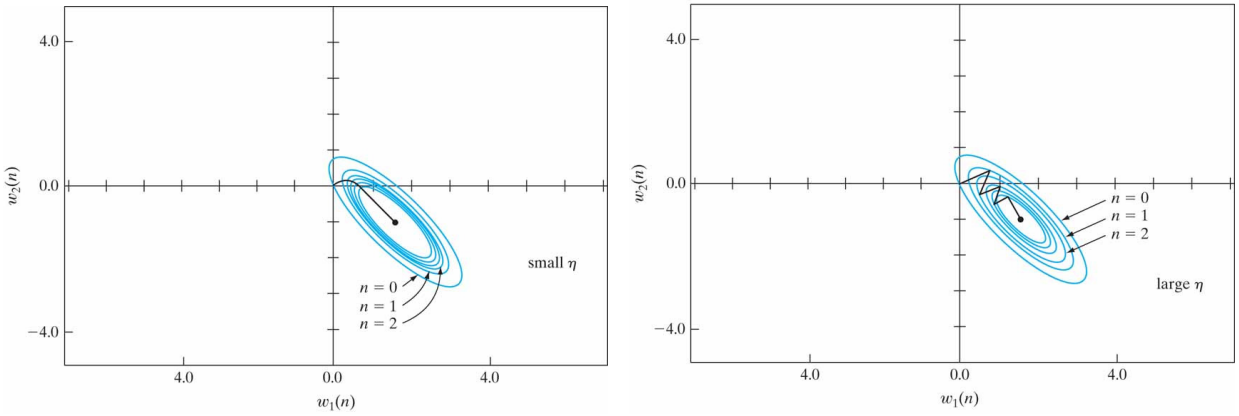
DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Entrenamiento de redes neuronales

Tasas de aprendizaje

Tasas de aprendizaje

Selección de la tasa de aprendizaje

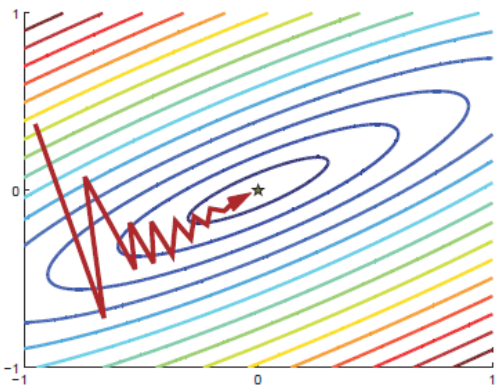


¿Cuánto se ajustan los pesos? Tasa de aprendizaje

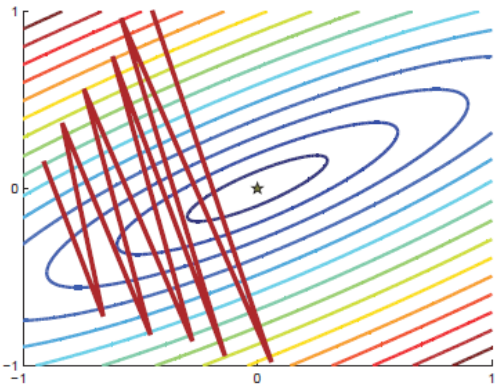
[Haykin: "Neural Networks and Learning Machines", 3rd edition]

Tasas de aprendizaje

Selección de la tasa de aprendizaje



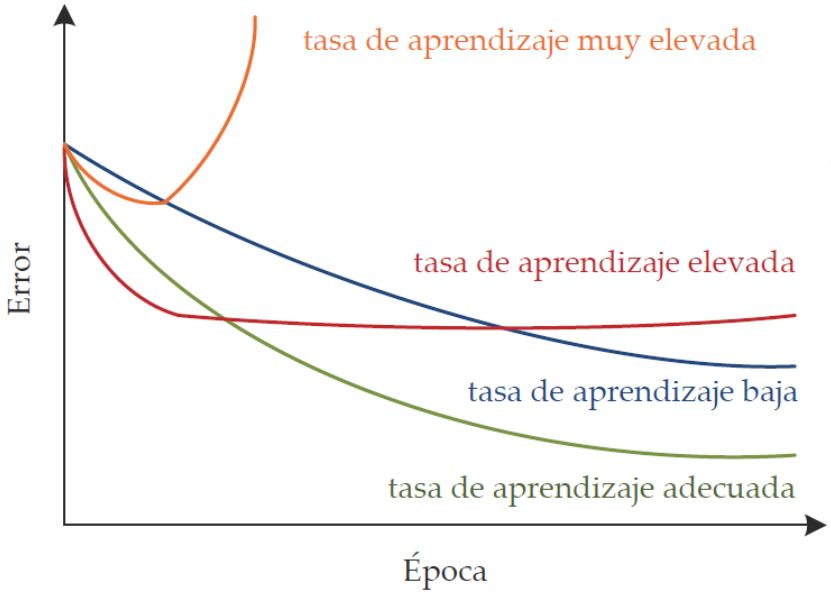
Tasa de aprendizaje adecuada (convergencia)

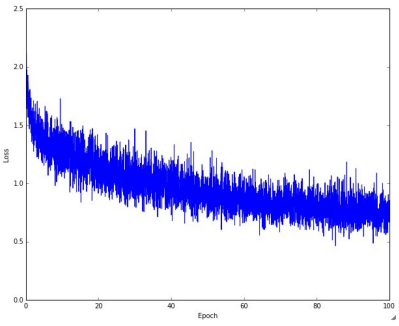


Tasa de aprendizaje demasiado alta (no convergencia)

Tasas de aprendizaje

Selección de la tasa de aprendizaje



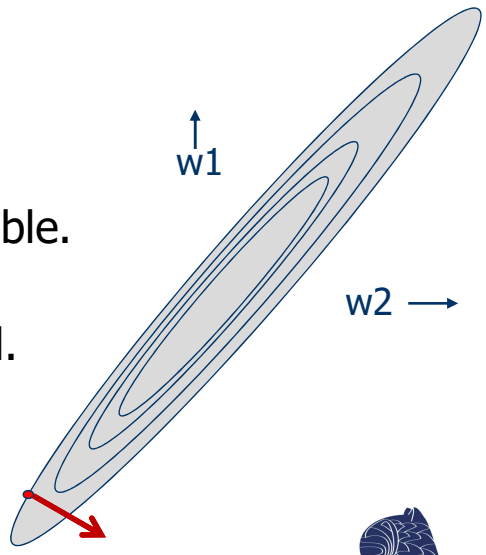


Tasas de aprendizaje

Tasa de aprendizaje

¿Cuánto se ajustan los pesos?

- Tasa global de aprendizaje fija.
- Tasa global de aprendizaje adaptable.
- Tasas de aprendizaje ajustadas para cada conexión/peso de la red.



Tasas de aprendizaje

Tasa de aprendizaje [LeCun et al.]
Consejos para la implementación de backpropagation

RECOMENDACIÓN: TASAS DE APRENDIZAJE / LEARNING RATES
Para igualar la velocidad de aprendizaje:

- Se le puede dar a cada peso su tasa de aprendizaje.
- Las tasas de aprendizaje deberían ser proporcionales a la raíz cuadrada del número de entradas de cada neurona.
- Los pesos de capas inferiores deberían ser mayores que los de capas superiores.

Tasas de aprendizaje

Tasa de aprendizaje [Hinton et al.]
usando mini-batch learning

Se estima una tasa de aprendizaje inicial:

- Si el error crece u oscila,
  se reduce la tasa de aprendizaje automáticamente.

- Si el error se va reduciendo de forma consistente pero lenta, se aumenta la tasa de aprendizaje.

Tasas de aprendizaje

Tasa de aprendizaje [Hinton et al.]
usando mini-batch learning

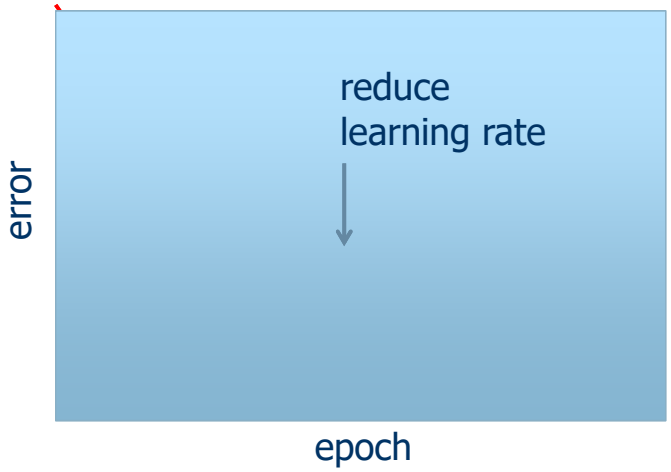
Al final del entrenamiento, casi siempre ayuda reducir la tasa de aprendizaje (se eliminan fluctuaciones en los pesos finales debidas a variaciones entre mini-lotes).

La tasa de aprendizaje se reduce si la estimación del error cometido deja de disminuir (estimación obtenida de un conjunto de validación distinto al de entrenamiento).

Tasas de aprendizaje

Tasa de aprendizaje [Hinton et al.]
usando mini-batch learning

¡Cuidado al reducir
la tasa de aprendizaje!



Reducir la tasa de aprendizaje reduce fluctuaciones aleatorias debidas a los distintos gradientes de los distintos mini-lotes, pero hace el aprendizaje más lento.

Tasas de aprendizaje

Tasa de aprendizaje [Hinton et al.]

PROBLEMA FRECUENTE

Si comenzamos con una tasa de aprendizaje demasiado elevada, los pesos de las neuronas ocultas adquirirán valores positivos muy altos o negativos muy bajos.

- Las derivadas del error para las neuronas ocultas serán minúsculas y el error no disminuirá.
- La saturación de las neuronas ocultas hace que nos quedemos estancados en una meseta [plateau], que se suele confundir con un mínimo local.

Tasas de aprendizaje

Ajuste de las tasas de aprendizaje

- Aproximación estocástica
  (disminución progresiva de la tasa de aprendizaje)

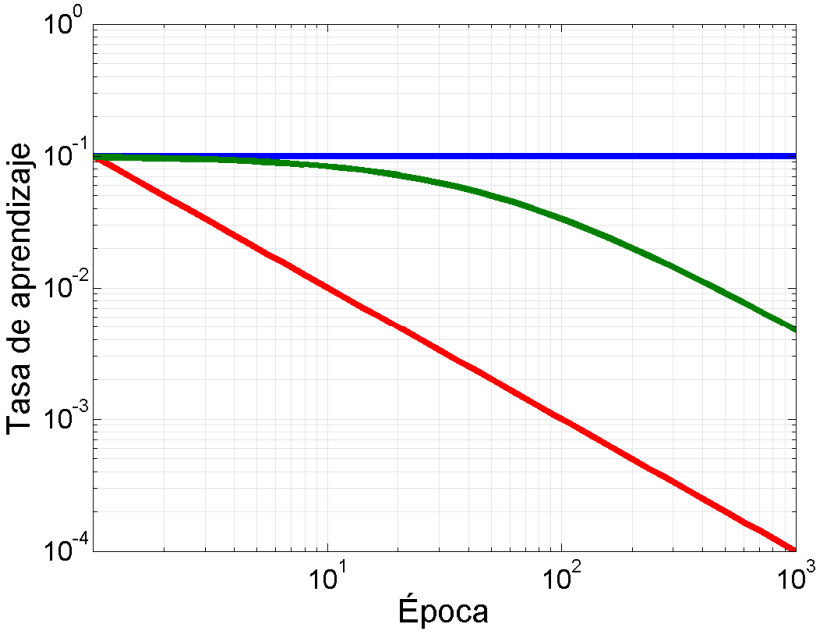
$$\eta(t) = \frac{\kappa}{t}$$

- Búsqueda y convergencia, p.ej. Keras
  [Darken & Moody, 1990]

$$\eta(t) = \frac{\eta_0}{1 + t/\tau}$$

Tasas de aprendizaje

Ajuste de las tasas de aprendizaje



Búsqueda y convergencia

Aproximación estocástica

Tasas de aprendizaje

Ajuste de las tasas de aprendizaje

Otras estrategias de ajuste

- **Power scheduling**
  (Darken & Moody con parámetro de potencia c)

$$\eta(t) = \eta_0 \,/\, (1 + t/s)^c$$

- **Exponential scheduling**
  (disminuye en un factor 10 cada s iteraciones)

$$\eta(t) = \eta_0 \, 0.1^{t/s}.$$

Tasas de aprendizaje

Ajuste de las tasas de aprendizaje

- **Performance scheduling**
  (se mide el error sobre un conjunto de validación cada N iteraciones y se reduce la tasa de aprendizaje por un factor λ si el error deja de disminuir)

- **1cycle scheduling**
  (se comienza aumentando linealmente la tasa de aprendizaje inicial η_0 hasta llegar a η_1 ($\approx 10\eta_0$) a mitad del entrenamiento; luego, se vuelve a reducir linealmente hasta η_0 y se sigue reduciendo varios órdenes de magnitud durante las últimas épocas).

Tasas de aprendizaje

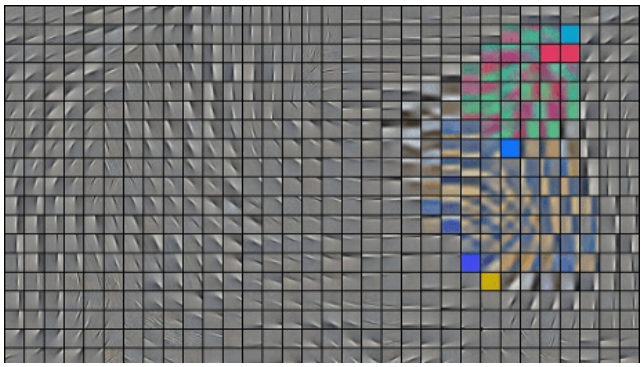
Ajuste de las tasas de aprendizaje

Una estrategia adecuada de ajuste de las tasas de aprendizaje puede ayudarnos a acelerar la convergencia del algoritmo de entrenamiento de una red neuronal.

La estrategia de Darken y Moody, así como sus derivadas más recientes (performance / exponential / 1cycle), reduce el número de épocas necesarias para ajustar los parámetros de la red.

Cursos

Neural Networks for Machine Learning
by Geoffrey Hinton
(University of Toronto & Google)
https://www.coursera.org/course/neuralnets



UNIVERSITY OF TORONTO

Cursos

Deep Learning Specialization
by Andrew Ng, 2017

- Neural Networks and Deep Learning
- Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
- Structuring Machine Learning Projects
- Convolutional Neural Networks
- Sequence Models

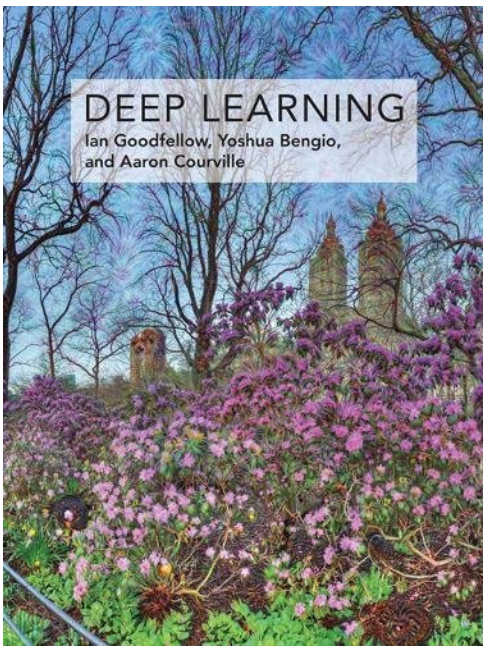
deeplearning.ai

https://www.coursera.org/specializations/deep-learning

Bibliografía

Lecturas recomendadas

Ian Goodfellow,
Yoshua Bengio
& Aaron Courville:
Deep Learning
MIT Press, 2016
ISBN 0262035618



http://www.deeplearningbook.org

Bibliografía

Lecturas complementarias

Fernando Berzal:
Redes Neuronales
& Deep Learning

CAPÍTULO 9
Entrenamiento de
una red neuronal