Redes Multicapa

Fernando Lozano

Universidad de los Andes

21 de agosto de 2014

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.
- Problema de asignación de crédito.

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.
- Problema de asignación de crédito.
- Algoritmos de entrenamiento más complejos

Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.
- Problema de asignación de crédito.
- Algoritmos de entrenamiento más complejos
- Dos clases más populares:

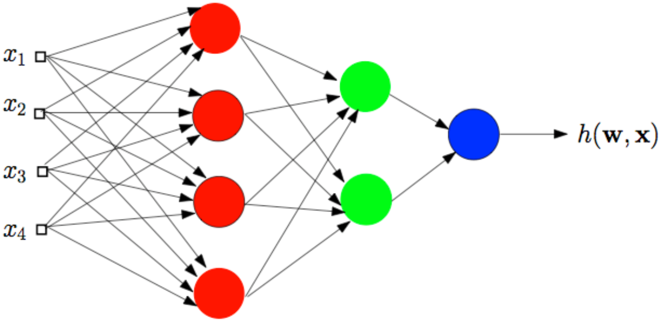
Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.
- Problema de asignación de crédito.
- Algoritmos de entrenamiento más complejos
- Dos clases más populares:
    - Perceptrón multinivel (Backpropagation).

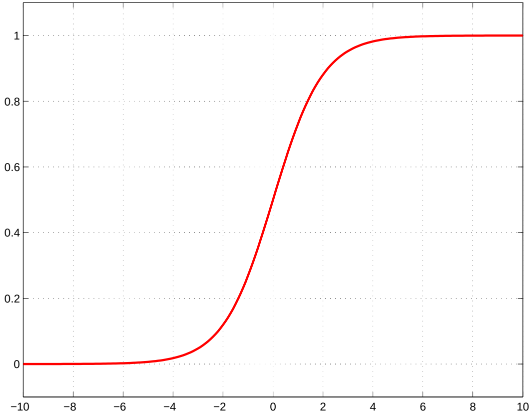
Redes Multicapa

- Redes con una sola capa tienen capacidad funcional limitada.
- Redes multicapa implementan funciones más complejas.
- Capacidad de construir representaciones internas de los datos de entrada.
- Problema de asignación de crédito.
- Algoritmos de entrenamiento más complejos
- Dos clases más populares:
  - Perceptrón multinivel (Backpropagation).
  - Funciones de base radial.

Arquitectura en Capas

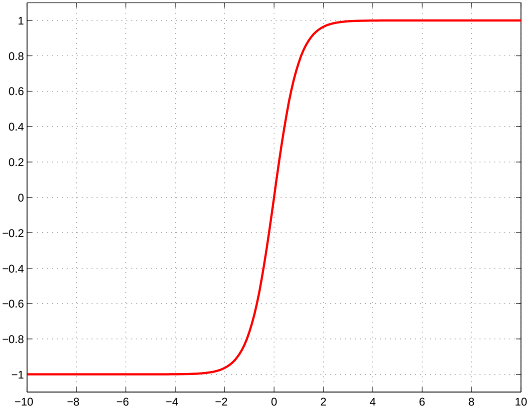


Activación sigmoidal



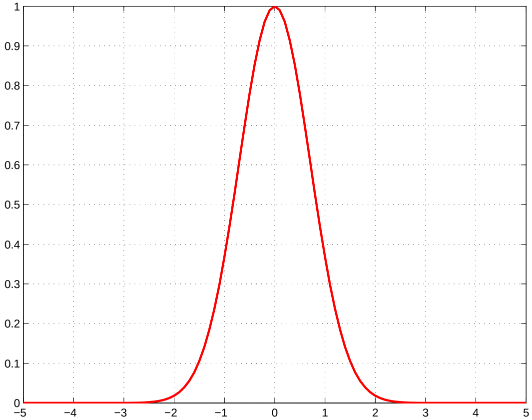
$$f_s(z) = \frac{1}{1 + e^{-\beta z}}$$

Tangente hiperbólica



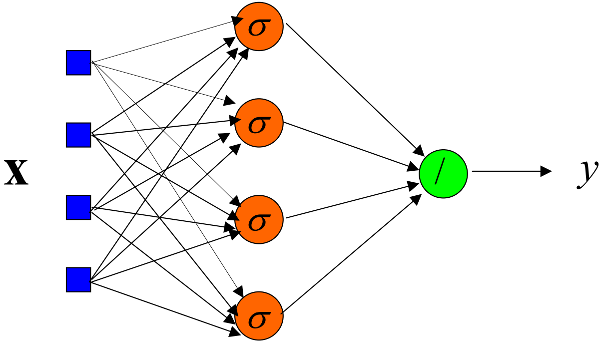
$$f_{TH}(z) = tanh(z)$$

Función de base radial

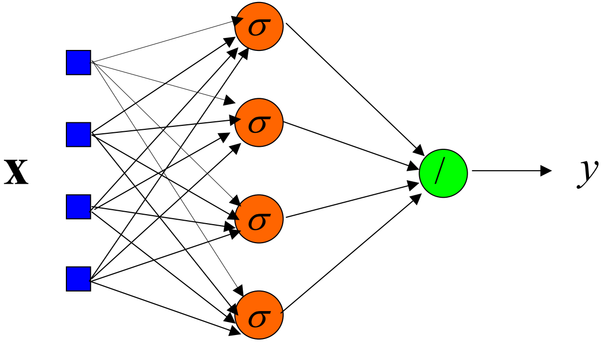


$$f_{RBF}(z) = e^{-z^2}$$

Perceptrón Multinivel (1 capa escondida)



Perceptrón Multinivel (1 capa escondida)



$$
\begin{aligned}
y = h(\mathbf{x}) &= f_o\left(a_0 + \sum_{k=1}^{N} a_k f_s(\mathbf{w}_k^T \mathbf{x})\right) \\
&= a_0 + \sum_{k=1}^{N} a_k f_s(\mathbf{w}_k^T \mathbf{x})
\end{aligned}
$$

Capacidad Funcional

Capacidad Funcional

- Aproximación universal : Una red con una capa escondida, es capaz de aproximar cualquier función "suave" con precisión arbitraria, si se permite incrementar el número de neuronas k indefinidamente.

Capacidad Funcional

- Aproximación universal : Una red con una capa escondida, es capaz de aproximar cualquier función “suave” con precisión arbitraria, si se permite incrementar el número de neuronas k indefinidamente.
  - MLPs: Cybenko (1989), Hornik (1991), Funahashi (1989)

Capacidad Funcional

- Aproximación universal : Una red con una capa escondida, es capaz de aproximar cualquier función “suave” con precisión arbitraria, si se permite incrementar el número de neuronas k indefinidamente.
  - MLPs: Cybenko (1989), Hornik (1991), Funahashi (1989).
  - RBFs: Hartman et.al (1990), Girossi y Pogio (1990).

Capacidad Funcional

- Aproximación universal : Una red con una capa escondida, es capaz de aproximar cualquier función "suave" con precisión arbitraria, si se permite incrementar el número de neuronas k indefinidamente.
  - MLPs: Cybenko (1989), Hornik (1991), Funahashi (1989).
  - RBFs: Hartman et.al (1990), Girossi y Pogio (1990).
- Esto no quiere decir que una red con una capa escondida y un número limitado de neuronas solucione cualquier problema!

Capacidad Funcional

- Aproximación universal : Una red con una capa escondida, es capaz de aproximar cualquier función "suave" con precisión arbitraria, si se permite incrementar el número de neuronas k indefinidamente.
  - MLPs: Cybenko (1989), Hornik (1991), Funahashi (1989).
  - RBFs: Hartman et.al (1990), Girossi y Pogio (1990).
- Esto no quiere decir que una red con una capa escondida y un número limitado de neuronas solucione cualquier problema!
- Tasa de aproximación $O\left(\frac{1}{n}\right)$ (es $O\left(\frac{1}{n^{\frac{1}{d}}}\right)$ para modelos lineales en los parámetros).

MLP: Algoritmo de entrenamiento

- Backpropagation: Werbos (1974), Rumelhart, Hinton, y Williams (1986), LeCun (1986), Parker (1985).

MLP: Algoritmo de entrenamiento

- Backpropagation: Werbos (1974), Rumelhart, Hinton, y Williams (1986), LeCun (1986), Parker (1985).
- Generalización del algoritmo LMS a múltiples capas.

MLP: Algoritmo de entrenamiento

- Backpropagation: Werbos (1974), Rumelhart, Hinton, y Williams (1986), LeCun (1986), Parker (1985).
- Generalización del algoritmo LMS a múltiples capas.
- Algoritmo iterativo de búsqueda usando gradientes.

MLP: Algoritmo de entrenamiento

- Backpropagation: Werbos (1974), Rumelhart, Hinton, y Williams (1986), LeCun (1986), Parker (1985).
- Generalización del algoritmo LMS a múltiples capas.
- Algoritmo iterativo de búsqueda usando gradientes.
- Regla de la cadena.

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w}, \mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w}, \mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w}, \mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:
     - Aleatorio.

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:
     - Aleatorio.
     - Aleatorio + normalización.

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:
     - Aleatorio.
     - Aleatorio + normalización.
     - Nguyen-Widrow.

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:
     - ★ Aleatorio.
     - ★ Aleatorio + normalización.
     - ★ Nguyen-Widrow.
  2. Búsqueda de gradiente: Ir "hacia abajo de la colina".

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_{\mathbf{w}} E|_{\mathbf{w}_k}$$

Backpropagation

- Función de error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w}, \mathbf{x}_p) - y_p)^2 = \frac{1}{2}\sum_{i=1}^{n} E_p$$

- Procedimiento iterativo:
  1. Punto inicial $\mathbf{w}_0$:
     - Aleatorio.
     - Aleatorio + normalización.
     - Nguyen-Widrow.
  2. Búsqueda de gradiente: Ir "hacia abajo de la colina".

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_{\mathbf{w}} E|_{\mathbf{w}_k}$$

- $\nabla_{\mathbf{w}} E|_{\mathbf{w}_k}$ puede calcularse exactamente (batch backpropagation), o estimarse con un sólo dato (on-line backpropagation).

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:

$$a_j = \sum_i w_{ji} z_i$$

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:
  $$a_j = \sum_i w_{ji} z_i$$
  Salida de neurona j:
  $$z_j = f_s(a_j)$$

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:
  $$a_j = \sum_i w_{ji} z_i$$
  Salida de neurona j:
  $$z_j = f_s(a_j)$$
- Usando la regla de la cadena:

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:
  $$a_j = \sum_i w_{ji} z_i$$
  Salida de neurona j:
  $$z_j = f_s(a_j)$$
- Usando la regla de la cadena:
  $$\frac{\partial Ep}{\partial w_{ji}} = \frac{\partial Ep}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:
  $$a_j = \sum_i w_{ji} z_i$$
  Salida de neurona j:
  $$z_j = f_s(a_j)$$
- Usando la regla de la cadena:
  $$\begin{aligned} \frac{\partial Ep}{\partial w_{ji}} &= \frac{\partial Ep}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \\ &= \delta_j z_i \end{aligned}$$

Cálculo del gradiente

$$f_s'(z) = f_s(z)(1 - f_s(z))$$

- Notaciones:
  Estado interno de neurona j:
  $$a_j = \sum_i w_{ji} z_i$$
  Salida de neurona j:
  $$z_j = f_s(a_j)$$
- Usando la regla de la cadena:
  $$\begin{aligned}\frac{\partial Ep}{\partial w_{ji}} &= \frac{\partial Ep}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \\ &= \delta_j z_i\end{aligned}$$

- Cálculo de δ_j:

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

  - En otro casos:

$$\delta_j = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

  - En otro casos:

$$\begin{aligned} \delta_j &= \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\ &= \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} f_s(a_i) \end{aligned}$$

- Cálculo de δ_j:
  - En la salida:
$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$
  - En otro casos:
$$\begin{aligned}\delta_j &= \sum_k \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial a_j} \\ &= \sum_k \delta_k \frac{\partial}{\partial a_j}\sum_i w_{ki} f_s(a_i) \\ &= f_s'(a_j)\sum_k w_{kj}\delta_k\end{aligned}$$

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

  - En otro casos:

$$\begin{aligned}
\delta_j &= \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\
&= \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} f_s(a_i) \\
&= f_s'(a_j) \sum_k w_{kj} \delta_k \\
&= f_s(a_j)(1 - f_s(a_j)) \sum_k w_{kj} \delta_k
\end{aligned}$$

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

  - En otro casos:

$$\begin{aligned}
\delta_j &= \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\
&= \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} f_s(a_i) \\
&= f_s'(a_j) \sum_k w_{kj} \delta_k \\
&= f_s(a_j)(1 - f_s(a_j)) \sum_k w_{kj} \delta_k \\
&= z_j(1 - z_j) \sum_k w_{kj} \delta_k
\end{aligned}$$

- Cálculo de δ_j:
  - En la salida:

$$\delta_j = h(\mathbf{w}, \mathbf{x}_p) - y_p$$

  - En otro casos:

$$\begin{aligned}
\delta_j &= \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\
&= \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} f_s(a_i) \\
&= f_s'(a_j) \sum_k w_{kj} \delta_k \\
&= f_s(a_j)(1 - f_s(a_j)) \sum_k w_{kj} \delta_k \\
&= z_j(1 - z_j) \sum_k w_{kj} \delta_k
\end{aligned}$$

Backpropagation

- Procedimiento iterativo.

Backpropagation

- Procedimiento iterativo.
- Dados valores de los pesos de la red, calcular la salida $h(\mathbf{w}, \mathbf{x}_i)$ (paso forward).

Backpropagation

- Procedimiento iterativo.
- Dados valores de los pesos de la red, calcular la salida $h(\mathbf{w}, \mathbf{x}_i)$ (paso forward).
- Calcular los δ_j desde la salida hacia las capas anteriores.

Backpropagation

- Procedimiento iterativo.
- Dados valores de los pesos de la red, calcular la salida $h(\mathbf{w}, \mathbf{x}_i)$ (paso forward).
- Calcular los δ_j desde la salida hacia las capas anteriores.El error se propaga desde la salida hacia las capas anteriores (paso de backpropagation).

Backpropagation

- Procedimiento iterativo.
- Dados valores de los pesos de la red, calcular la salida $h(\mathbf{w}, \mathbf{x}_i)$ (paso forward).
- Calcular los δ_j desde la salida hacia las capas anteriores.El error se propaga desde la salida hacia las capas anteriores (paso de backpropagation).
- Actualizar los pesos.

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$
  Feed-forward $\{$calcule los $z_j\}$

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$
  Feed-forward {calcule los z_j}
  Back-prop {calcule los δ_j y $\nabla_{\mathbf{w}} E_p$}

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$
  Feed-forward {calcule los z_j}
  Back-prop {calcule los δ_j y $\nabla_{\mathbf{w}} E_p$}
  $\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_{\mathbf{w}} E_p \big|_{\mathbf{w}_k}$

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$
  Feed-forward {calcule los z_j}
  Back-prop {calcule los δ_j y $\nabla_{\mathbf{w}} E_p$}
  $\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_{\mathbf{w}} E_p \big|_{\mathbf{w}_k}$
until Condición de terminación.

Algoritmo de Backpropagation (on-line)

Incialize $\mathbf{w}_0$
repeat
  Escoja $(\mathbf{x}_p, y_p)$
  Feed-forward {calcule los z_j}
  Back-prop {calcule los δ_j y $\nabla_{\mathbf{w}} E_p$}
  $\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla_{\mathbf{w}} E_p \big|_{\mathbf{w}_k}$
until Condición de terminación.

Eficiencia de Backpropagation

Eficiencia de Backpropagation

- Sea W el número de pesos en la red.

Eficiencia de Backpropagation

- Sea W el número de pesos en la red.
- Cálculo en pasos forward y de backpropagation toma $O(W)$ operaciones.

Eficiencia de Backpropagation

- Sea W el número de pesos en la red.
- Cálculo en pasos forward y de backpropagation toma $O(W)$ operaciones.
- Cálculo directo toma

Eficiencia de Backpropagation

- Sea W el número de pesos en la red.
- Cálculo en pasos forward y de backpropagation toma $O(W)$ operaciones.
- Cálculo directo toma $O(W^2)$ operaciones.

Eficiencia de Backpropagation

- Sea W el número de pesos en la red.
- Cálculo en pasos forward y de backpropagation toma $O(W)$ operaciones.
- Cálculo directo toma $O(W^2)$ operaciones.
- Multiplicar por n datos!

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).
  2. Gradiente aproximado (versión on-line).

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).
  2. Gradiente aproximado (versión on-line).
- Consideramos la función:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{Q}(\mathbf{w} - \mathbf{w}^*), \qquad \mathbf{Q} > 0$$

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).
  2. Gradiente aproximado (versión on-line).
- Consideramos la función:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{Q}(\mathbf{w} - \mathbf{w}^*), \qquad \mathbf{Q} > 0$$

(buena aproximación cerca al mínimo local $\mathbf{w}^*$, si suponemos $E(\mathbf{w}^*) = 0$)

Velocidad de Convergencia

- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).
  2. Gradiente aproximado (versión on-line).
- Consideramos la función:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{Q}(\mathbf{w} - \mathbf{w}^*), \qquad \mathbf{Q} > 0$$

  (buena aproximación cerca al mínimo local $\mathbf{w}^*$, si suponemos $E(\mathbf{w}^*) = 0$)
- Convergencia de steepest descent depende del número de condición de $\mathbf{Q}$: $\quad \kappa = \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}}$:

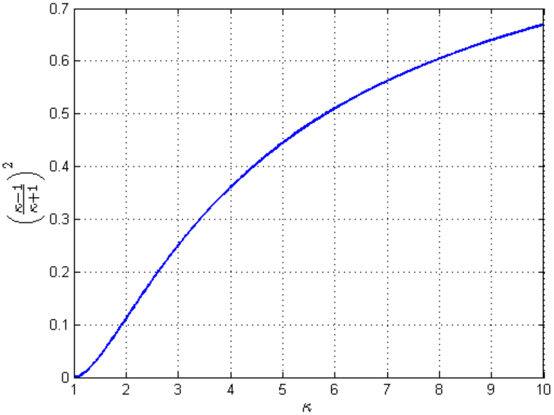
Velocidad de Convergencia

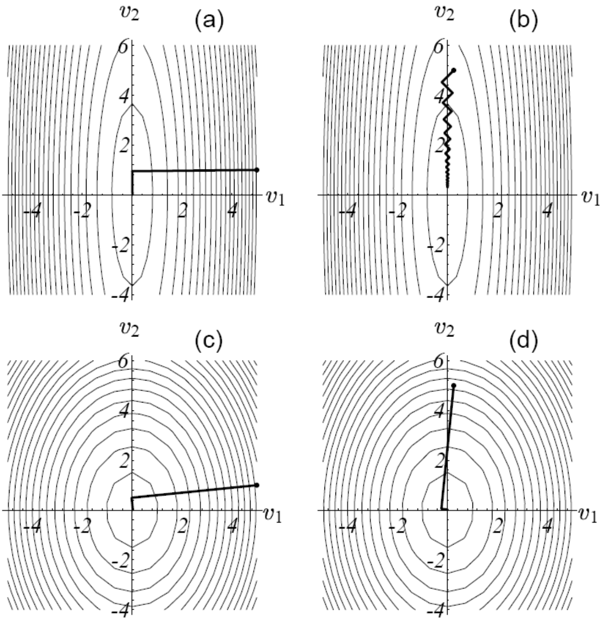
- Backpropagation es una versión simplificada de steepest descent:
  1. Tasa de aprendizaje (no hay búsqueda de línea).
  2. Gradiente aproximado (versión on-line).
- Consideramos la función:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{Q}(\mathbf{w} - \mathbf{w}^*), \qquad \mathbf{Q} > 0$$

  (buena aproximación cerca al mínimo local $\mathbf{w}^*$, si suponemos $E(\mathbf{w}^*) = 0$)
- Convergencia de steepest descent depende del número de condición de $\mathbf{Q}$: $\kappa = \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}}$:

$$E(\mathbf{w}_{k+1}) \leq E(\mathbf{w}_k) \left(\frac{\kappa - 1}{\kappa + 1}\right)^2$$


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
κ
$\left(\frac{\kappa-1}{\kappa+1}\right)^2$


(a)
(b)
(c)
(d)
v_1
v_2
6
4
2
-2
-4

Variaciones de Backpropagation

- Heurísticas:

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:
  - Dirección de búsqueda

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:
  - Dirección de búsqueda
    - Quasi Newton

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:
  - Dirección de búsqueda
    - Quasi Newton
    - Gradiente Conjugado

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:
  - Dirección de búsqueda
    - Quasi Newton
    - Gradiente Conjugado
    - Levenberg-Marquardt.

Variaciones de Backpropagation

- Heurísticas:
  - Momentum.
  - Tasa de aprendizaje variable.
  - Backpropagation resistente (resilient).
- Técnicas de Optimización:
  - Dirección de búsqueda
    - Quasi Newton
    - Gradiente Conjugado
    - Levenberg-Marquardt.
  - Técnicas de búsqueda de línea.

Método de Levenberg-Marquardt

- Función de error:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 \\
&= \frac{1}{2}\sum_{i=1}^{n} e^2 = \frac{1}{2}\|\mathbf{e}\|^2
\end{aligned}
$$

Método de Levenberg-Marquardt

- Función de error:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p)-y_p)^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}e^2 = \frac{1}{2}\|\mathbf{e}\|^2
\end{aligned}
$$

- Si $\mathbf{w}_n - \mathbf{w}_v$ es pequeño:

$$
\mathbf{e}(\mathbf{w}_n) \approx \mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)
$$

Método de Levenberg-Marquardt

- Función de error:

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n} e^2 = \frac{1}{2}\|\mathbf{e}\|^2 \end{aligned}$$

- Si $\mathbf{w}_n - \mathbf{w}_v$ es pequeño:

$$\mathbf{e}(\mathbf{w}_n) \approx \mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)$$

donde

$$(\mathbf{J})_{ni} = \frac{\partial e_n}{\partial w_i}$$

Método de Levenberg-Marquardt

- Función de error:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{p=1}^{n}(h(\mathbf{w},\mathbf{x}_p) - y_p)^2 \\
&= \frac{1}{2}\sum_{i=1}^{n} e^2 = \frac{1}{2}\|\mathbf{e}\|^2
\end{aligned}
$$

- Si $\mathbf{w}_n - \mathbf{w}_v$ es pequeño:

$$
\mathbf{e}(\mathbf{w}_n) \approx \mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)
$$

donde

$$
(\mathbf{J})_{ni} = \frac{\partial e_n}{\partial w_i}
$$

- La función de error:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2$$

- La función de error:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2$$

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- La función de error:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2$$

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Los elementos de la matriz Hessiana son:

$$(\mathbf{H})_{ik} = \frac{\partial^2 E}{\partial w_i \partial w_k}$$

- La función de error:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2$$

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Los elementos de la matriz Hessiana son:

$$\begin{aligned}(\mathbf{H})_{ik} &= \frac{\partial^2 E}{\partial w_i \partial w_k} \\ &= \sum_n \left\{ \frac{\partial e_n}{\partial w_i}\frac{\partial e_n}{\partial w_k} + e_n \frac{\partial^2 e_n}{\partial w_i \partial w_k} \right\}\end{aligned}$$

- La función de error:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2$$

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Los elementos de la matriz Hessiana son:

$$\begin{aligned}(\mathbf{H})_{ik} &= \frac{\partial^2 E}{\partial w_i \partial w_k} \\ &= \sum_n \left\{ \frac{\partial e_n}{\partial w_i}\frac{\partial e_n}{\partial w_k} + e_n \frac{\partial^2 e_n}{\partial w_i \partial w_k} \right\}\end{aligned}$$

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$$

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$$

- Para una red lineal, esta aproximación es exacta.

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$$

- Para una red lineal, esta aproximación es exacta.
- En esta aproximación la Hessiana es relativamente fácil de calcular usando backpropagation.

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$$

- Para una red lineal, esta aproximación es exacta.
- En esta aproximación la Hessiana es relativamente fácil de calcular usando backpropagation.
- En Levenberg-Marquardt se minimiza la función de error modificada:

$$E = \frac{1}{2}\|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2 + \lambda\|\mathbf{w}_n - \mathbf{w}_o\|^2$$

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$$

- Para una red lineal, esta aproximación es exacta.
- En esta aproximación la Hessiana es relativamente fácil de calcular usando backpropagation.
- En Levenberg-Marquardt se minimiza la función de error modificada:

$$E = \frac{1}{2} \|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2 + \lambda \|\mathbf{w}_n - \mathbf{w}_o\|^2$$

- La idea es tratar de minimizar el error cuadrático medio, limitando al mismo tiempo el tamaño del paso, de manera que la aproximación sea válida.

- Cerca al mínimo de la función de error, el segundo término en la sumatoria se puede despreciar, y entonces podemos escribir la Hessiana como:

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$$

- Para una red lineal, esta aproximación es exacta.
- En esta aproximación la Hessiana es relativamente fácil de calcular usando backpropagation.
- En Levenberg-Marquardt se minimiza la función de error modificada:

$$E = \frac{1}{2} \|\mathbf{e}(\mathbf{w}_o) + \mathbf{J}(\mathbf{w}_n - \mathbf{w}_o)\|^2 + \lambda \|\mathbf{w}_n - \mathbf{w}_o\|^2$$

- La idea es tratar de minimizar el error cuadrático medio, limitando al mismo tiempo el tamaño del paso, de manera que la aproximación sea válida.

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Para valores pequeños de λ se tiene método de Newton.

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Para valores pequeños de λ se tiene método de Newton.
- Para valores grandes de λ se tiene descenso de gradiente con paso $1/\lambda$.

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Para valores pequeños de λ se tiene método de Newton.
- Para valores grandes de λ se tiene descenso de gradiente con paso $1/\lambda$.
- Usualmente λ se adapta durante el proceso de optimización:

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Para valores pequeños de λ se tiene método de Newton.
- Para valores grandes de λ se tiene descenso de gradiente con paso $1/\lambda$.
- Usualmente λ se adapta durante el proceso de optimización:
  - Si el error decrece, λ se multiplica por 10.

- minimizando con respecto a $\mathbf{w}_n$:

$$\mathbf{w}_n = \mathbf{w}_o - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}_o)$$

- Para valores pequeños de λ se tiene método de Newton.
- Para valores grandes de λ se tiene descenso de gradiente con paso $1/\lambda$.
- Usualmente λ se adapta durante el proceso de optimización:
  - Si el error decrece, λ se multiplica por 10.
  - Si el error aumenta, se descarta el vector de pesos λ divide por 10.