

# Redes neuronales multicapa

Castiglione, Karpovsky, Sturla

Sistemas de Inteligencia Artificial

3 de Mayo de 2012

## 1 Introducción

- El problema

## 2 Modelado del problema

- Representación de la red neuronal
- Funciones de activación
- Arquitecturas
- Cálculo del error
- Conjuntos de entrenamiento y testeo

## 3 Mejoras al algoritmo backpropagation

- Eta dinámico
- Ruido y momentum

## 4 Resultados

## 5 Conclusiones

# Tabla de contenidos

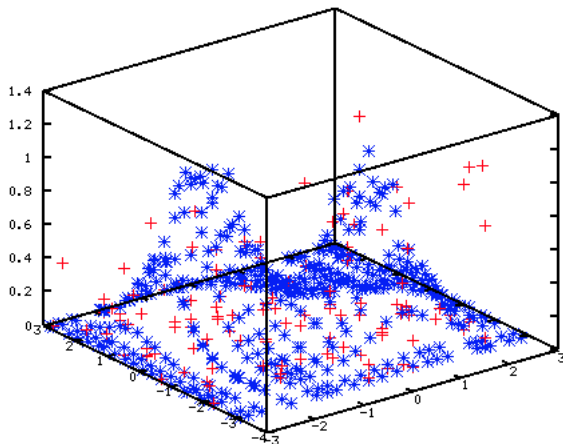
- 1 **Introducción**
  - El problema
- 2 **Modelado del problema**
  - Representación de la red neuronal
  - Funciones de activación
  - Arquitecturas
  - Cálculo del error
  - Conjuntos de entrenamiento y testeo
- 3 **Mejoras al algoritmo backpropagation**
  - Eta dinámico
  - Ruido y momentum
- 4 **Resultados**
- 5 **Conclusiones**

# El problema

El problema planteado consiste en la estimación de funciones escalares a partir de un conjunto de puntos que las representan.

En nuestro caso particular hemos trabajado con el archivo **`samples7.txt`**

# Gráfico de la función



# Tabla de contenidos

- 1 Introducción
  - El problema
- 2 **Modelado del problema**
  - Representación de la red neuronal
  - Funciones de activación
  - Arquitecturas
  - Cálculo del error
  - Conjuntos de entrenamiento y testeo
- 3 Mejoras al algoritmo backpropagation
  - Eta dinámico
  - Ruido y momentum
- 4 Resultados
- 5 Conclusiones

# Representación de la red neuronal

Se representó la red neuronal como una matriz de pesos.

- Cada neurona es una columna de pesos.
- Cada capa de neuronas es una matriz de pesos.
- La red neuronal, por consiguiente, es un vector de matrices.

Se utilizaron dos funciones de activación distintas:

### Sigmoidea exponencial

$$g(h) = \frac{1}{1 + e^{-2\beta h}}$$

Derivada:

$$2\beta g(1 - g)$$

### Tangente hiperbólica

$$g(x) = \tanh(x)$$

Derivada:

$$\beta g(1 - g^2)$$



# Arquitecturas estudiadas

- Perceptron simple?
- Pocas neuronas, pocas capas
- Muchas neuronas, muchas capas
- Punto intermedio
- Conexiones muertas/rotas

# Conjunto de entrenamiento y testeo

Se decidió seguir el consejo de la cátedra y al realizar las pruebas se utilizó un subconjunto de los datos seleccionados al azar para la fase de aprendizaje y el subconjunto restante para testeo.

- Elección de puntos al azar? Puntos representativos?

# Tabla de contenidos

- 1 Introducción
  - El problema
- 2 Modelado del problema
  - Representación de la red neuronal
  - Funciones de activación
  - Arquitecturas
  - Cálculo del error
  - Conjuntos de entrenamiento y testeo
- 3 Mejoras al algoritmo backpropagation
  - Eta dinámico
  - Ruido y momentum
- 4 Resultados
- 5 Conclusiones

## Eta dinámico ( $\eta$ adaptativo)

Incrementar  $\eta$  si el error sube consistenemente incrementar eta aritméticamente: quizás se está siendo muy conservador.  
Si el error incrementa, reducir  $\eta$  exponencialmente.

# Momentum

$$w_{ij}(t+1) = -\frac{\partial E}{\partial w_{ij}} + \alpha w(t)$$

Cambios dependen de cambios anteriores. Idea de dirección general del error.

Olvido exponencial de cambios anteriores.

Se aplica a cada batch / lote

# Ruido

Idea: Escape del mínimo local.

Robustez de la red neuronal: debería poder soportar ruido.

# Tabla de contenidos

- 1 Introducción
  - El problema
- 2 Modelado del problema
  - Representación de la red neuronal
  - Funciones de activación
  - Arquitecturas
  - Cálculo del error
  - Conjuntos de entrenamiento y testeo
- 3 Mejoras al algoritmo backpropagation
  - Eta dinámico
  - Ruido y momentum
- 4 Resultados
- 5 Conclusiones

## Resultados 1

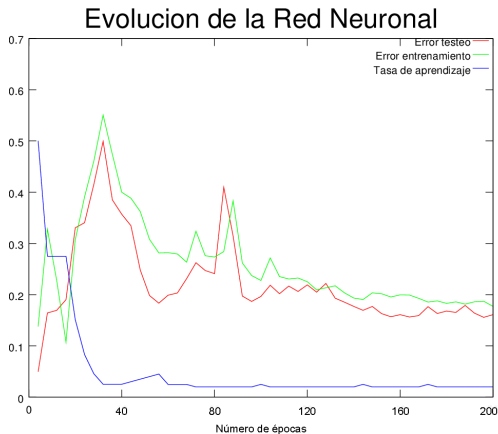


Figura 1: Arq [200 100], eta adaptativo, tangente hiperbólica



## Resultados 2

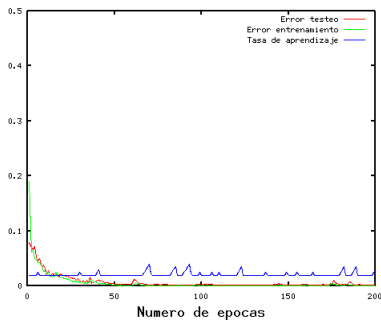


Figura 2: Arq [50 30] 200 epocas, tangente hiperbólica.

## Resultados 3

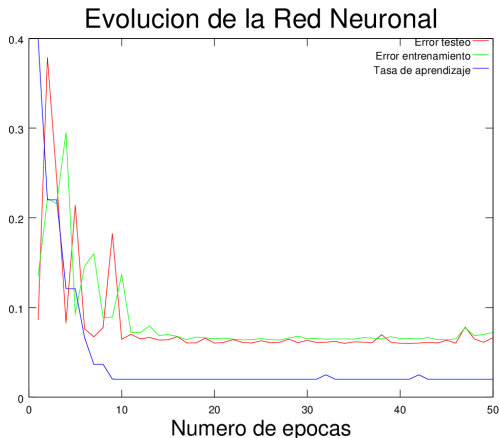


Figura 3: Arq [4 4] 200 épocas eta adaptativo, tanh.

# Tabla de contenidos

- 1 Introducción
  - El problema
- 2 Modelado del problema
  - Representación de la red neuronal
  - Funciones de activación
  - Arquitecturas
  - Cálculo del error
  - Conjuntos de entrenamiento y testeo
- 3 Mejoras al algoritmo backpropagation
  - Eta dinámico
  - Ruido y momentum
- 4 Resultados
- 5 Conclusiones

# Conclusión

- Incrementar la cantidad de neuronas arbitrariamente no necesariamente implica mejoras en cuanto al error (puede llevar a malas generalizaciones y tiempo de más hasta alcanzar el error desdado). Esto puede observarse en la figura 1 del **Anexo A**
- No existe tal cosa como una mejor arquitectura o parámetros óptimos. Estos seguramente dependan el problema que se está analizando.

## Conclusión cont.

- Momentum no siempre puede acelerar la convergencia.
- La función de activación *exponencial* es más propensa a atascarse en mínimos.
- Tomar pocos puntos puede ser una muestra poco representativa, y por lo tanto, puede haber mala generalización.