



Aprendizaje Supervisado

Diplomado en Computación Inteligente, UPB 2005

Jerónimo Castrillón Mazo
IEO UPB.
www.geocities.com/jeronimocm





Previa... Saludo

- **Enfoque:** vista desde adentro... vamos a aprender a hacer no a aplicar... ¿otro extremo?
- Concepto general de Asup... cualquier estructura se puede aprender... cuestión de entender y saber derivar.
- Una NN es un **Aprox. De func.** En el fondo también lo es un sistema difuso y otras estructuras.
- Concepto general de aprendizaje.
- Creencias sobre *Backprop*. Lo de ahora, SVM, De. Falta conocimiento.
- Falla... falta de contacto con lo real, tiempo de preparación!
- **HORARIO**
- **CALIFICACIÓN:** Cod., toolbox matlab, LevMarqAlg, DB (*benchmark*).
- Pag web. Presentación. Recomendación para hoy.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com



2



Contenido

- Introducción
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*



Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 3



Introducción

- Tipos de aprendizaje
- Historia
- Objetivo
- Visión general
- Modelo general de una Neurona

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 4

Tipos de Aprendizaje



Imagine un organismo o máquina que experimenta una serie de entradas sensoriales :

$$x_1, x_2, \dots, x_n$$

Dependiendo de la información adicional y de lo que se espera que el agente realice existen 4 tipos de aprendizaje.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

5

Tipos de Aprendizaje (2)



- **Aprendizaje supervisado:** al agente se le entrega también **salidas deseadas:**

$$d_1, d_2, \dots, d_n$$

y debe aprender a producir salidas correctas ante nuevas entradas.
- **Aprendizaje no supervisado:** El objetivo del agente es construir **representaciones** de x que sean útiles para procesos de decisión, predicción, reconocimiento, razonamiento, comunicación, etc.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

6

Tipos de Aprendizaje (3)

- **Aprendizaje por refuerzo:** el agente produce también acciones:



$$a_1, a_2, \dots, a_n$$

que modifican el estado del ambiente y recibe recompensas (o castigos):

$$r_1, r_2, \dots, r_n$$

Su objetivo es aprender a actuar de tal manera que maximice la recompensa a largo plazo.
- **Aprendizaje con índice de desempeño:** El agente recibe una medida cuantitativa por su comportamiento.



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
7

Historia

- Golgi y Ramon y Cajal, siglo XIX: estudian el sistema nervioso y descubren las neuronas.
- McCulloch y Pitts, 1943: primer red neuronal artificial, binaria.
- Hebb, 1949: *Hebbian learning*. Correlación entre estímulo-respuesta. Aprendizaje no supervisado.
- Minsky, 1954: Aplica redes neuronales a aprendizaje por refuerzo.
- Rosenblatt, 1958: aparece el *perceptron*, una sola neurona con aprendizaje supervisado.



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
8

Historia (2)

- Widrow y Hoff, 1960: *Adaline: ADaptive LINar combiner Element*.
- Grossberg, 80's: *adaptive resonance theory* (ART). Toda una familia de algoritmos de aprendizaje no supervisado (Más sobre esto en el módulo de ANS).
- Hopfield, 80's: *Hopfield network*.
- Kohonen, 82, 89: *self-organising feature map* (SOM). (Más sobre esto en el módulo de ANS.)
- Oja, 1982: *neural principal component analysis* (PCA). (Más sobre esto en el módulo de ANS.)
- Rumelhart, Hinton and Williams: *backpropagation*

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 9

Historia (3)

Lo último... años 90's:

- *Machine learning*: mucho rigor matemático. Análisis de convergencia y determinación de cotas. Reglas de Bayes, SVM (*Support Vector Machines*). **Mi Exp.**
- Modelos biológicos para implementar en HW, *spiking neurons* (Wulfram Gerstner and Werner M. Kistler, 2002).

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 10



Visión General

Las redes neuronales artificiales (RNA o ANN) se aplican a:

- Clasificación
- Reconocimiento de patrones (caras, voz, letra manuscrita)
- Regresión, aproximación de funciones
- Predicción (e. g. bolsa)

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

11



Visión General (2)

- Se tiene una base de datos, *training set*, donde se almacena la información a ser aprendida.
- Una NN esta especificada por:
 - Arquitectura: dada por un conjunto de unidades (neuronas) y un conjunto de conexiones. Cada conexión posee un peso.
 - Modelo neuronal.
 - Algoritmo de aprendizaje.
- El propósito es lograr una red con la habilidad de *generalización*.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

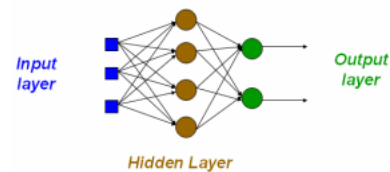
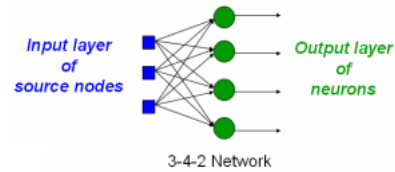
12



Visión General (3)

Tipos de arquitecturas

- *Feed-forward*: de conexiones hacia delante, *single-layer* o *multi-layer*.



Feb-Jun, 2005

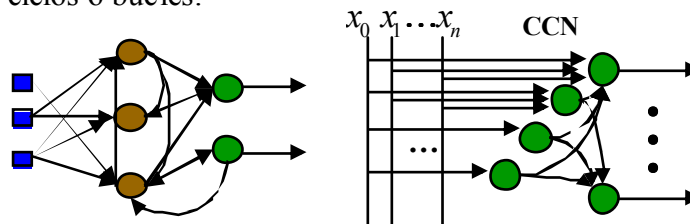
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

13



Visión General (4)

- Recurrentes: conectadas de cualquier forma. Puede haber ciclos o bucles.



Nota: la arquitectura que se utilice está directamente relacionada con el algoritmo de aprendizaje.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

14



Visión General (5)

Algoritmo de aprendizaje:

- Adapta los pesos en las conexiones para resolver la tarea particular de aprendizaje.
- Dos modos fundamentales: *Batch*, *Incremental*.
- Una red neuronal es una función paramétrica. Usualmente el problema de entrenamiento se afronta como un problema de optimización de alguna función, conocida como *criterion function* (función objetivo) $J(w)$.
- Tasa de aprendizaje: concepto muy importante. Normalmente un número positivo que regula la **velocidad** y la **estabilidad** del sistema.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

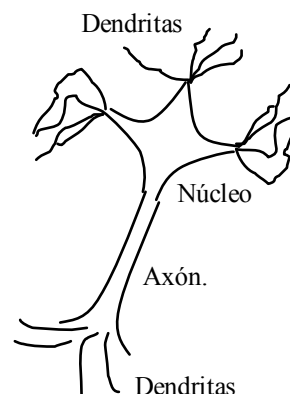
15



Modelo General de una Neurona

Biológicamente (*a grandes rasgos*):

- Impulsos nerviosos entran por las dendritas ubicadas alrededor del núcleo.
- Las sinapsis regulan el efecto de impulsos pre-sinápticos en la pos-sinapsis, e.g. la conductividad.
- Si los pulsos superan un determinado umbral la neurona produce un impulso nervioso por el axón, el cual a su vez se propaga a otras neuronas a través de las dendritas inferiores



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

16



Modelo Neuronal (2)

Hechos:

- El ser humano posee del orden de 10^{11} neuronas. Un caracol 9.
- El modelamiento biológico es muy complicado. El algorítmico mucho más sencillo.
- La codificación biológica se encuentra en la frecuencia de los pulsos, en la fase de los mismo y en otros factores.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

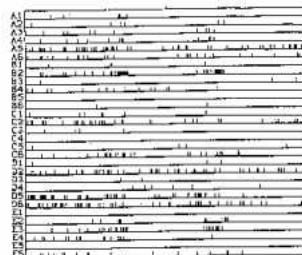
17



Modelo Neuronal (3)



This reproduction of a drawing of Ramón y Cajal shows a few neurons in the mammalian cortex that he observed under the microscope. Only a small portion of the neurons contained in the sample of cortical tissue have been made visible by the staining procedure; the density of neurons is in reality much higher. Cell b is a nice example of a pyramidal cell with a triangular shaped cell body. Dendrites, which leave the cell laterally and upwards, can be recognized by their rough surface. The axons are recognizable as thin, smooth lines which extend downwards with a few branches to the left and right. From Ramón y Cajal (1909).





Spatio-temporal pulse pattern. The spikes of 30 neurons (A1-E6, plotted along the vertical axis) are shown as a function of time (horizontal axis, total time is 4000 ms). The firing times are marked by short vertical bars. From Krüger and Aiple (1983).

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

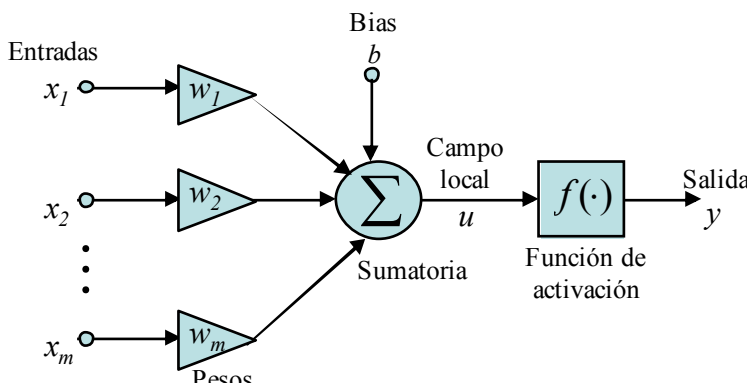
18





Modelo Neuronal (4)


- Modelo simplificado:




Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

19





Modelo Neuronal (5)

La neurona básica esta descrita entonces por:

- Un conjunto de conexiones caracterizadas por unos pesos, w_j que regulan el efecto de la entrada sobre la neurona.
- Un sumador que asemeja la “acumulación de voltaje” al interior de núcleo:

$$u = b + \sum_{j=1}^m w_j \cdot x_j = b + \vec{w}^T \cdot \vec{x}$$

- Bias* o polarización, b , que representa un nivel presente en el núcleo independiente de la entrada. Se acostumbra modelar el *bias* como una entrada $x_0=1$ conectada con un peso $w_0=b$.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

20



Modelo Neuronal (6)

- Una función de activación, $f(u)$ (también *squashing function*), que regula la salida de la neurona a través del axón. La salida es entonces:

$$y = f(u) = f\left(\sum_{j=0}^m w_j \cdot x_j\right) = f\left(w_0 + \vec{w}^T \cdot \vec{x}\right)$$

- Note que una NN representa en últimas una función parametrizada. El problema se reduce a encontrar el vector de pesos que mejor resuelva el problema. El como hallar dicho vector \vec{w}^* es el propósito de todos los algoritmos de aprendizaje.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

21




Contenido

- [Introducción](#)
- ***Perceptron learning Rule***
- *Adaline* (α -LMS)
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

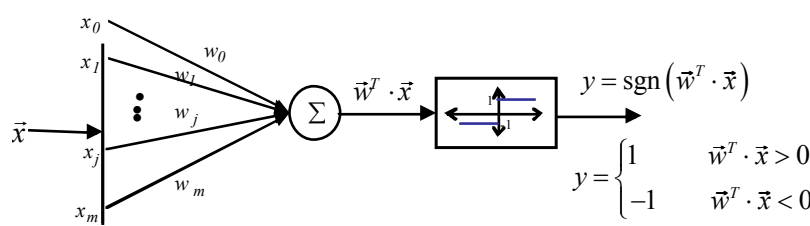
22




Perceptron learning Rule




- Rosenblatt, 1958. Es el más sencillo de todos.
- Neurona cuya función de activación es la función *signo*.
- Mapea un vector $\vec{x} = [x_0 \ x_2 \ \dots \ x_m]^T \in R^{m+1}$ a una salida bipolar. Es un clasificador simple de 2 clases.
- Bias*: $x_0 = 1$



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
23



Perceptron (2)



- Considere el *training set*: $\{X, D\}$. Donde X es la matriz con las entradas $x^k \in R^{m+1}$ y D con la salida deseada $d^k \in \{-1, +1\}$. (Por simplicidad de notación se obvia la flecha.) Objetivo: hallar w^* tal que:

$$\begin{cases} w^{*T} \cdot x^k > 0 & \text{si } d^k = +1 \\ w^{*T} \cdot x^k < 0 & \text{si } d^k = -1 \end{cases} \quad \forall x^k \in X$$
- Observe que $w^{*T} \cdot x$ define un hiperplano en $R^m \Rightarrow$
Geoméricamente la labor de un perceptrón es hallar un hiperplano que realice una partición del espacio, de manera que una región queden todos los patrones para los cuales d^k sea +1 y en la otra los de -1.

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
24



Perceptrón: Entrenamiento

- Una de las formas para entrenar esta unidad, Rosenblat 1962:

$$\begin{cases} \text{iniciar } w^1 \text{ de manera arbitraria} \\ w^{k+1} = w^k + \alpha (d^k - y^k) x^k, \quad k = 1, 2, \dots \end{cases}$$

donde:

- k representa la iteración o el momento en el cual se selecciona la entrada. Por lo general se necesitan varios ciclos sobre la base de datos. Si ésta es finita con N datos y k se utiliza para indexar los patrones en el *training set* se debe utilizar $((k-1) \bmod N) + 1$.
- $0 < \alpha$ es una constante llamada tasa (tasa) de aprendizaje (*learning rate*).
- y^k es la salida del perceptrón ante la entrada x^k .

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

25



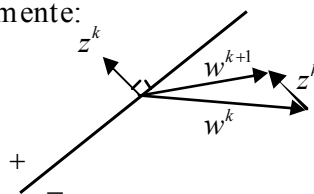
Perceptrón: Entrenamiento (2)

- Note que si $\alpha = 0.5$:

$$\begin{cases} w^{k+1} = w^k + z^k & \text{si } (w^k)^T \cdot z^k \leq 0 \\ w^{k+1} = w^k & \text{de otra manera} \end{cases} \quad \text{con } z^k = \begin{cases} +x^k & \text{si } d^k = +1 \\ -x^k & \text{si } d^k = -1 \end{cases}$$

Misclassification


- Entonces su busca solucionar: $(w^k)^T \cdot z^k > 0 \quad \forall k$
- Geométricamente:




Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

26



Perceptrón: Ejemplo 2D




- Tablero. Geometría de la corrección. Importancia del *bias*.
- AND, OR.
- Famoso problema de la XOR


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

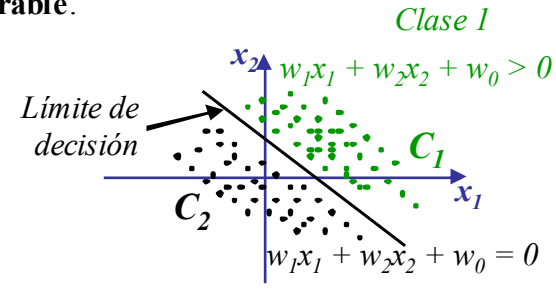
27



Perceptrón: Análisis



- Para analizar: Existencia y Convergencia de solución
- Existencia: obliga a que el *set* de datos sea **linealmente separable**.



Clase 1

$w_1x_1 + w_2x_2 + w_0 > 0$

$w_1x_1 + w_2x_2 + w_0 = 0$

- Convergencia. Novikoff 1962, Nilsson 1965.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

28



Perceptrón: Convergencia

$$w^{*T} \cdot z^k > 0 \quad \forall k = 1, 2, 3, \dots$$

$$w^{k+1} - \rho w^* = w^k - \rho w^* + z^k$$

$$\|w^{k+1} - \rho w^*\|^2 = \|w^k - \rho w^*\|^2 + 2(w^k - \rho w^*)^T z^k + \|z^k\|^2$$

$$\|w^{k+1} - \rho w^*\|^2 \leq \|w^k - \rho w^*\|^2 - 2\rho w^{*T} z^k + \|z^k\|^2, \text{ ya que } (w^k)^T \cdot z^k \leq 0$$

$$\|w^{k+1} - \rho w^*\|^2 \leq \|w^k - \rho w^*\|^2 - 2\rho\gamma + \beta^2, \text{ con } \gamma = \min_i (w^{*T} z^i) > 0,$$

$$\beta^2 = \max_i \|z^i\|^2$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

29



Perceptrón: Convergencia (2)

Escogiendo ρ grande, $\rho = \beta^2 / \gamma$:

$$\|w^{k+1} - \rho w^*\|^2 \leq \|w^k - \rho w^*\|^2 - \beta^2, \text{ y luego de correcciones:}$$

$$0 \leq \|w^{k+1} - \rho w^*\|^2 \leq \|w^1 - \rho w^*\|^2 - k\beta^2$$

$$\Rightarrow k_0 = \frac{\|w^{k+1} - \rho w^*\|^2}{\beta^2}$$

Por lo tanto, si la solución existe, se alcanzará antes de este máximo de iteraciones. Note que depende del vector de pesos óptimo, desconocido (típico con los *Bounds*).

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

30



Perceptrón: Variaciones

- Rata de aprendizaje variable con las iteraciones y un margen positivo fijo b (*Margin*). (Dibujo en tablero)

$$\begin{cases} w^{k+1} = w^k + \alpha^k z^k & \text{si } (w^k)^T \cdot z^k \leq b \\ w^{k+1} = w^k & \text{de otra manera} \end{cases}$$

- *Batch update*: más rápido, sin pruebas serias de convergencia.

$$w^{k+1} = w^k + \alpha \sum_{z \in Z(w^k)} z$$

Donde $Z(w^k)$ es el set de patrones mal clasificados por w^k en una corrida de la base de datos.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

31



Perceptrón: Variaciones (2)


- Butz rule, 1967: trata de evitar el problemas del perceptrón en ciertas aplicaciones. Modifica el peso aun cuando el patrón este bien clasificado (añade un factor de *refuerzo*).
- May's learning rule, 1964:

$$\begin{cases} w^{k+1} = w^k + \alpha^k \frac{b - (w^k)^T \cdot z^k}{\|z^k\|^2} & \text{si } (w^k)^T \cdot z^k \leq b \\ w^{k+1} = w^k & \text{de otra manera} \end{cases}$$


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

32




Contenido




- Introducción
- *Perceptron learning Rule*
- **Adaline (α -LMS)**
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*

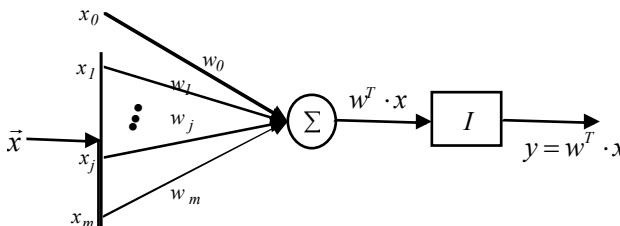
Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
33



Adaline (α -LMS)



- Widrow-Hoff, 1960.
- *Adaline: ADaptive LINear combiner Element.*
- Función de activación: función identidad.
- Además de clasificación de datos no necesariamente linealmente separables permite hacer regresión.



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
34



Adaline (2)

- En el *training set* $d^k \in R$.
- La regla de aprendizaje se conoce como: α -LMS,

$$\begin{cases} w^1 = 0 \text{ o arbitraria} \\ w^{k+1} = w^k + \alpha (d^k - y^k) \frac{x^k}{\|x^k\|^2} \end{cases}$$

- *Minimal disturbance principle*: nuevos patrones cambian poco lo aprendido con otros patrones diferentes, debido a que el *update* se hace colinear con x^k .

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

35




Adaline: Comparación

- Perceptrón:
 - Una sola capa.
 - Modelo neuronal no lineal.
 - Clasificación
- Adaline:
 - Una sola capa
 - Modelo neuronal lineal
 - Clasificación y regresión.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

36




Contenido

- [Introducción](#)
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- **Descenso de gradiente (*gradient descent*)**
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

37



Descenso de Gradiente

- Nombre completo: *steepest gradient descent search rule*. O simplemente: *gradient descent*.
- Hasta ahora las reglas que se han visto son heurísticas.
- La mayoría de los algoritmos hacen descenso de gradiente. MUY IMPORTANTE (algunos combinan con Hessiano).
- Hecho: El gradiente de una función escalar apunta en la dirección de máximo crecimiento. En 2D es la pendiente de la recta tangente.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

38



Descenso de Gradiente (2)

- Las reglas de aprendizaje se basan en descender sobre una superficie de error hasta un mínimo (ojalá global!).
Dependiendo de la función de error o *criterion function* ($J(w)$) se tienen una u otra regla de aprendizaje.
- Si $J(w)$ es una función de clase 1, se pueden hallar los puntos estacionarios w^* ($\nabla J(w^*) = \vec{0}$) de manera analítica. Sistema de ecuaciones $(m+1)(m+1)$ difícil de resolver (más aun con funciones de activación no lineales).

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

39



Descenso de Gradiente (3)

- La idea: actualizar los datos de manera que estos se muevan en la dirección de decrecimiento de $J(w)$. La fórmula general que gobierna el descenso de gradiente es:

$$w^{k+1} = w^k + \alpha A \nabla J(w) \Big|_{w=w^k} = w^k + \alpha A \left[\frac{\partial J}{\partial w_0} \quad \frac{\partial J}{\partial w_1} \quad \dots \quad \frac{\partial J}{\partial w_m} \right]^T \Big|_{w=w^k}$$


$\alpha \in R, A \in M_{m+1}$, usualmente: $A = I_{m+1}$

- $\alpha > 0$ hace descenso de gradiente. $\alpha < 0$ ascenso.
- En cualquier caso se puede llegar a un punto silla. Bien por el ruido.
- Si $A = H(J(w))^{-1} = (\nabla \nabla J(w))^{-1}$ y $\alpha = 1 \Rightarrow$ Newton.


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

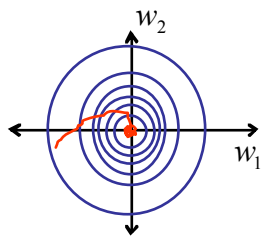
40



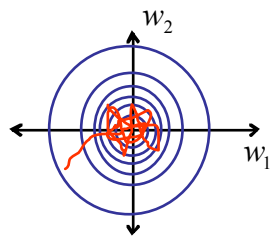
Descenso de Gradiente (4)



- El valor de α es muy importante. Puede acelerar el aprendizaje pero también puede causar oscilaciones alrededor del óptimo.



α "apropiada"




α "grande"

MOSTRAR 2D


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

41



DG: *batch* vs *incremental*



Hasta ahora, modo *batch*.

Ventajas:

- Bajo simples condiciones garantiza convergencia.
- Muchos trucos para acelerar (ej. Segundo orden LM Alg.)



Desventajas:

- Lento: todo un ciclo antes de actualizar. BD grandes y redundantes.
- Supone un *training set* disponible. En muchas aplicaciones, sin embargo, un *training set* no está constituido a priori, sino que va construyéndose de manera iterativa.
- Necesita grandes recursos de memoria.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

42



DG: *batch* vs *incremental* (2)



Modo incremental:

- El modo incremental hace descenso de gradiente sobre una muestra del error, e.g. a cada paso de aprendizaje.
- También: *descenso de gradiente estocástico*. Por que los gradientes a cada paso no necesariamente apuntan en la misma dirección del gradiente *batch*... pero efecto final similar.

Ventajas:

- Apropiado para aplicaciones *on-line*.
- Más rápido, sobre todo en base de datos grandes y con redundancia.
- Las trayectorias estocásticas permiten escapar de mínimos locales (ILUSTRAR).

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 43



DG: *batch* vs *incremental* (3)

Desventajas (incremental):

- Inestable a no ser que la tasa de aprendizaje se decremente.
- No posee pruebas teóricas de convergencia tan buenas como para *batch* – son probabilísticas.
- La mayoría de los métodos de segundo orden no aplican.

No obstante las desventajas, el modo incremental es el más utilizado

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 44



DG: *batch vs incremental* (4)

Ilustración: velocidad del incremental

- BD con 1000 muestras. 100 diferentes repetidas 10 veces.
- BATCH: no aprovecha la redundancia. El promedio de 10 datos iguales es igual al error en uno solo.
- INCREMENTAL: actualiza con cada repetición.

→ ***batch* será como mínimo 10 veces más lento**

En la vida real no se repiten muestras, pero si se presentan redundancias, datos parecidos.

OPCIÓN: *MiniBatches*.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

45



DG: Perceptrón

- La regla heurística del perceptrón se puede plantear partiendo de un descenso de gradiente de la función objetivo (Duda y Hart 1973):

$$J(w) = - \sum_{z \in Z(w)} z^T \cdot w$$

Observe que si: $Z(w) = \emptyset \Rightarrow J(w) = 0$

de otra manera: $J(w) > 0$ dado que $z^T \cdot w \leq 0$ por estar mal clasificado.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

46



DG: Perceptrón (2)

- El gradiente es: (DEMOSTRARLO, TABLERO)

$$\nabla J(w) = - \sum_{z \in Z(w)} z \Rightarrow w^{k+1} = w^k + \alpha \sum_{z \in Z(w^k)} z$$

Lo mismo que antes!, para la regla *batch*.

De igual forma, utilizando *margin* y para la regla de May:

$$J(w) = - \sum_{z^T w \leq b} (z^T \cdot w - b) \quad J(w) = \frac{1}{2} \sum_{z^T w \leq b} \frac{(z^T \cdot w - b)^2}{\|z\|^2}$$

Por qué positivo??

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

47



DG: Perceptrón (3)

$$\nabla J(w) = \nabla \left(- \sum_{z \in Z(w)} z^T \cdot w \right) = \nabla \left(- \sum_{k=1}^K (z^k)^T \cdot w \right) (*) \longrightarrow \text{Por facilidad}$$

$$\nabla J(w) = - \left[\frac{\partial}{\partial w_0} \left(\sum_{k=1}^K (z^k)^T \cdot w \right), \frac{\partial}{\partial w_1} \left(\sum_{k=1}^K (z^k)^T \cdot w \right), \dots, \frac{\partial}{\partial w_m} \left(\sum_{k=1}^K (z^k)^T \cdot w \right) \right]$$

$$\nabla J(w) = - \left[\frac{\partial}{\partial w_0} \left(\sum_{k=1}^K \sum_{i=0}^m z_i^k \cdot w_i \right), \dots, \frac{\partial}{\partial w_m} \left(\sum_{k=1}^K \sum_{i=0}^m z_i^k \cdot w_i \right) \right]$$

$$\nabla J(w) = - \left[\sum_{k=1}^K \frac{\partial}{\partial w_0} \sum_{i=0}^m z_i^k \cdot w_i, \dots, \sum_{k=1}^K \frac{\partial}{\partial w_m} \sum_{i=0}^m z_i^k \cdot w_i \right]$$


$$\nabla J(w) = - \left[\sum_{k=1}^K z_0^k, \sum_{k=1}^K z_1^k, \dots, \sum_{k=1}^K z_m^k \right] = - \left[z_0^1 + z_0^2 + \dots + z_0^K, \dots, z_m^1 + z_m^2 + \dots + z_m^K \right]$$

$$\nabla J(w) = - \left(\left[z_0^1, z_1^1, \dots, z_m^1 \right] + \left[z_0^2, z_1^2, \dots, z_m^2 \right] + \dots + \left[z_0^K, z_1^K, \dots, z_m^K \right] \right) = - \sum_{j=1}^K z$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com


48



Contenido

- [Introducción](#)
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*


Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 49




μ -LMS y *Delta rule*

- Las reglas que aparecieron luego se derivaron de una manera sistemática, definiendo primero la función objetivo y luego desarrollando para hallar la regla de aprendizaje.
- μ -LMS: Widrow y Hoff (1960). Similar a α -LMS.
- Ahora si: LMS = *least mean squares*.
- μ -LMS, junto con su extensión la regla delta, es el método de aprendizaje más utilizado y analizado.
- Fácilmente extensible a más capas.
- Inicialmente, para una unidad lineal.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 50



μ-LMS



- Función objetivo: suma del error cuadrático, SSE (*Sum of Squared Error*). Esto se hace sobre los N datos del set:

$$SSE(w) = J(w) = \frac{1}{2} \sum_{j=1}^N (d^j - y^j)^2, \text{ con } y^j = (x^j)^T \cdot w$$
- Aplicando el descenso de gradiente:

$$\nabla_w J(w) = \frac{1}{2} \nabla_w \sum_{j=1}^N (d^j - (x^j)^T \cdot w)^2 = \frac{1}{2} 2 \sum_{j=1}^N (d^j - (x^j)^T \cdot w) (-x^j)$$


$$\nabla_w J(w) = - \sum_{j=1}^N (d^j - y^j) x^j$$

➔


$$w^{k+1} = w^k + \alpha \sum_{j=1}^N (d^j - y^j) x^j$$

Batch LMS rule

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
51




μ-LMS (2)




- Note que el SSE es una función cuadrática en y .
- y mantiene una relación lineal con w .
- SSE es cuadrático en los pesos. Define un hiperparaboloide, superficie convexa que solo posee un mínimo!
- μ-LMS, con una tasa de aprendizaje decreciente, converge asintóticamente al peso óptimo w^* .
- Regla μ-LMS incremental: $w^{k+1} = w^k + \alpha (d^k - y^k) x^k$

igual a α-LMS si se hace: $\alpha = \alpha^k = \frac{\alpha_{\alpha-LMS}}{\|x^k\|^2}$

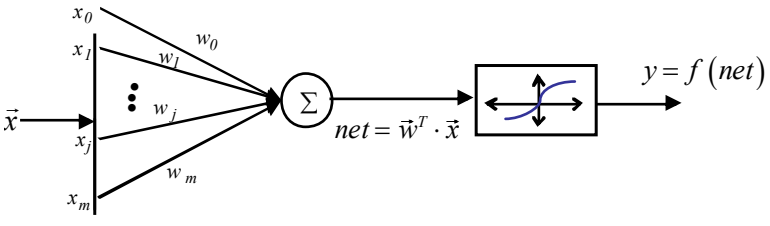
Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
52



Delta Rule




- Extensión de μ -LMS a neuronas con funciones de activación diferenciable.




- Entradas $x^k \in \mathbb{R}^{m+1}$ con: $x_0^k = 1 \forall k$ y $d^k \in [-1, +1]$

$\Rightarrow \nabla J(w) = -(d - y) \cdot f'(net) \cdot x$

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
53



Delta Rule (2)



- Lo anterior lleva a la regla delta de aprendizaje:

$$w^{k+1} = w^k + \alpha (d^k - f(net^k)) f'(net^k) x^k,$$

con $net^k = (x^k)^T \cdot w^k$

- Esta regla es de las más utilizadas... fácil extensión a varias capas.
- Para clasificación se usan normalmente, como funciones de activación:

Tangente hipérbolica: $f(x) = \tanh(\beta x) \Rightarrow f'(x) = \beta(1 - f^2(x))$

Función logística: $f(x) = 1/(1 + e^{-\beta x}) \Rightarrow f'(x) = \beta f(x)(1 - f(x))$

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
54



Delta Rule (3)

- Para clasificación se ubica un umbral.
- El hecho de que sea continua permite analizar que tan bien clasificado esta.
- Desventaja: *flat spots*. Hacen lento el aprendizaje al correr las iteraciones. Se puede corregir sumando un *bias*:

$$w^{k+1} = w^k + \alpha (d^k - f(net^k)) (f'(net^k) + \varepsilon) x^k$$

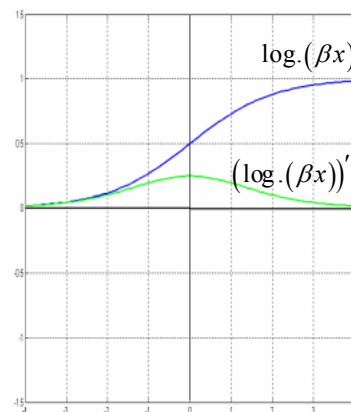
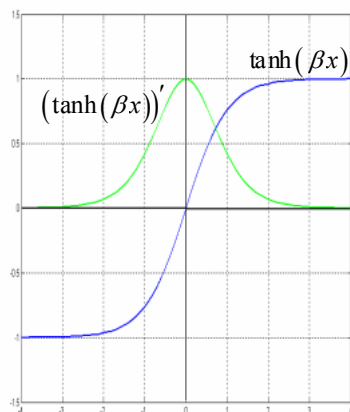
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

55



Delta Rule (4)



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

56



Palabras Finales: Reglas de Aprendizaje

- Existen muchas otras que dependen de la función objetivo:
 - AHK 1,2,3: Hassoun y Song 1992.
 - *Minkowsky-r criterion*: escala r^{-1} y suma de las potencias r de los valores absolutos (función signo en el gradiente).
 - *Maximun likelihood, log likelihood*.
 - Baum, Wilczek 1988: Medida del error de la entropía relativa instantánea (Kullback, 1959).
- Extensión a unidades estocásticas.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

57





Contenido

- Introducción
- *Perceptron learning Rule*
- *Adaline* (α -LMS)
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- **Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)**
- Mejoras a *backprop*
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

58



Redes Multicapa

- Extensión de la regla delta a redes multicapa.
- El algoritmo resultante: *backpropagation* o *backprop*. Algunos libros en español, *retropropagación*.
- Para redes *Feed-Forward* (FFNN).
- Muy utilizado e importante. Fácil y “eficiente”.
- Utilizada para: reconocimiento de patrones, aprox. de funciones, modelamiento de sistemas no lineales (control), predicción, compresión y reconstrucción de imágenes.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

59

Redes Multicapa (2)

Consideremos una red de dos capas, la primera con J unidades y la segunda con L :


- Entrada a la red: $x = [x_0 = 1, x_1, x_2, \dots, x_n] \in R^{n+1}$
- Salida de la red: $y = [y_1, y_2, \dots, y_L] \in R^L$
- Salida deseada: $d = [d_1, d_2, \dots, d_L] \in R^L$
- Salida de la primera capa: $z = [z_1, z_2, \dots, z_J] \in R^J$
- Entrada a la segunda capa: $z = [z_0 = 1, z_1, z_2, \dots, z_J] \in R^{J+1}$

La última capa se llama capa de salida. Las demás se llaman capas escondidas.


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

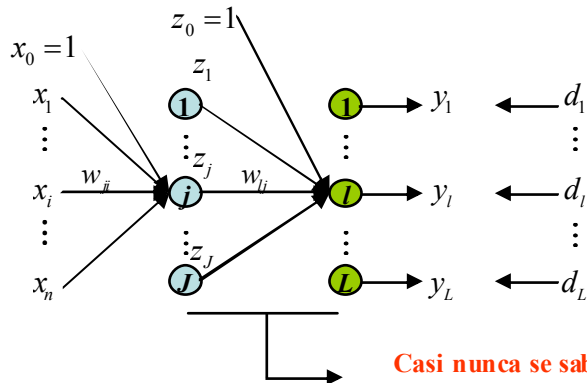
60



Redes Multicapa (3)




Red de dos capas totalmente interconectada hacia adelante




Resuelve problemas no sepables linealmente!!

Casi nunca se sabe cuantas unidades se necesitan!

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
61



Redes Multicapa (4)



- f_h , función de activación de la capa escondida es usualmente una función no lineal, sigmoideal.
- f_o , función de la capa de salida puede ser lineal o sigmoideal dependiendo de la aplicación.
- Por lo general no se utilizan diferentes funciones.
- Los parámetros de las sigmoideales se ajustan cercano a 1.
- Si tanto f_h como f_o son lineales, se puede colapsar a una sola capa, cuya matriz de pesos es el producto de las dos anteriores.
- Una red con capa inicial no lineal es un aproximador universal.
- No es “normal” utilizar más de dos capas.
- $J(n+1)+L(J+1)$ parámetros a optimizar.

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
62



Redes Multicapa (5)

- Si las funciones son diferenciables entonces se puede hacer descenso de gradiente sobre alguna función objetivo.
- Se puede usar cualquier función objetivo.
- Varios investigadores llegaron de manera independiente al análisis con el SSE, entre ellos: Parker, 1985.
- Si se utiliza como *criterion function* una muestra instantánea del SSE (Rumelhart *et al.*, 1986):

$$E(w) = J(w) = \frac{1}{2} \sum_{l=1}^L (d_l^k - y_l^k)^2 \quad \text{con } y^k = f_o \left((z^k)^T \cdot w \right)$$

Se deriva la **Error Backpropagation Learning Rule**.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

63



Backpropagation

- Para los pesos de las unidades de la salida se puede aplicar directamente la regla delta:

$$\Delta w_{lj} = (w_{lj})^n - (w_{lj})^c = -\alpha_o \frac{\partial J(w)}{\partial w_{lj}} = \alpha_o (d_l - y_l) f'_o(net_l) z_j,$$


$$\text{con } net_l = \sum_{j=0}^J w_{lj} z_j, \quad l = 1, 2, \dots, L, \quad j = 0, 1, \dots, J$$

- Para las demás capas el error se *propaga* hacia atrás, aplicando la regla de la cadena.


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

64



Backpropagation (2)

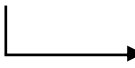


- El cambio para un peso en la primera capa es:

$$\Delta w_{ji} = -\alpha_h \frac{\partial J(w)}{\partial w_{ij}}, j = 1, 2, \dots, J, i = 0, 1, \dots, n$$

de donde:


$$\Delta w_{ji} = \alpha_h \left[\sum_{l=1}^L (d_l - y_l) f'_o(net_l) w_{lj} \right] f'(net_j) x_i$$

 **DEMOSTRARLO**


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

65



Backpropagation (3)



- DEMOSTRACIÓN SEGUNDA CAPA.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

66



Backpropagation (4)

- Observe que para la segunda capa se puede definir un *target* así:

$$d_j \triangleq z_j + \sum_{l=1}^L (d_l - y_l) f'_o(\text{net}_l) w_{lj}$$

- Además si se trabaja con funciones como la tangente hiporbólica o la función logística, las derivadas se pueden expresar en la forma:

$$f(\text{net}) = \tanh(\beta \text{net}) \Rightarrow f'(\text{net}) = \beta(1 - f^2(\text{net}))$$

$$f(\text{net}) = \frac{1}{(1 + e^{-\beta \text{net}})} \Rightarrow f'(\text{net}) = \beta f(\text{net})(1 - f(\text{net}))$$

Feb-Jun, 2005

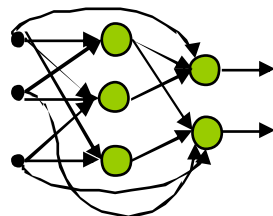
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

67



Backpropagation (5)

- INTERESANTE:** Siempre que se plantee una arquitectura con funciones de activación diferenciables, se puede intentar derivar ecuaciones de actualización utilizando un procedimiento similar. Pudiéndose entrenar arquitecturas con conexiones diferentes (p.e. un sistema difuso):




<Si se realizan conexiones con ciclos, aparecerán ecuaciones diferenciales de actualización>


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

68



Backpropagation (6)




- Procedimiento *backprop incremental*:
 - Inicio de los pesos.
 - Inicio de las tasas de aprendizaje (α_o , α_h).
 - Selecciona x^k del *training set*, preferiblemente de manera aleatoria. Propagar la entrada por la red para obtener y^k .
 - Tomar d^k de la base de datos y realizar las actualizaciones. Se puede hacer la de salida primero y luego la escondida ... pero significa más gasto computacional.
 - Revisar convergencia, criterio de parada (todo un tema: *overfitting*, *cross-validation*, ...).


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

69



Contenido





- Introducción
- *Perceptron learning Rule*
- *Adaline* (α -LMS)
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- **Mejoras a *backprop***
- RBF: *radial basis function*
- Otras
- SVM: *support vector machines*

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

70





Backprop: Mejoras y Variaciones

- Este algoritmo sufre de varios problemas y de temas sin resolver por completo.
- Velocidad y efectividad (Mínimos locales).
- Inicialización (preprocesamiento, pesos, tasas de aprendizaje) y terminación (Criterios de paro).

En las siguientes diapositivas se mencionan algunas mejoras y variaciones que buscan disminuir el efecto de estos problemas.

Backprop ha sido muy estudiado y trabajado, por lo tanto existen muchas mejoras documentadas.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 71



BP: Mejoras y Variaciones (2)

- Velocidad y Efectividad: Léon Bottou: “*BackProp is a simple algorithm, but convergence can take ages if it is not used properly*”.
 - La superficie de error de una red multicapa no es cóncava ni convexa. NO EXISTE una técnica para hallar el mínimo global... se debe emplear métodos heurísticos.
 - Pequeños detalles de implementación pueden variar el tiempo de convergencia en ordenes de magnitud.
 - OJO: en problemas de la vida real BackProp funciona mejor que en pequeños problemas!

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 72



BP: *Shuffling examples*

- En cualquier momento, seleccionar el patrón con mayor contenido de información.
 - El que posee mayor error.
 - El que más se diferencia a sus predecesores.
- Simple: Desordenar (*shuffle*) la base de datos para que puntos consecutivos difícilmente pertenezcan a la misma clase. Para gradiente estocástico para clasificación.
- Refinado, *Emphasizing Scheme*: entrenar con patrones *difíciles* más a menudo que con patrones fáciles (en términos de error). Problemas:
 - Perturba la distribución de las entradas
 - Tiempo de cómputo
 - *Outliers*

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

73



BP: Sigmoide

- Utilizar sigmoides simétricas, p.e. tangente hiperbólica.
- Regla: La media de las entradas a una neurona debe tener una media pequeña comparada con la desviación estándar.
- Problema: *flat spots**. Adicionar una función lineal de pendiente pequeña, es decir, sumar una constante pequeña a la derivada. Existen otras opciones, p.e. actualizar dinámicamente el parámetro β para ajustar la pendiente de cada unidad o utilizar funciones no sigmoideas (polinómicas, racionales ... **se sabe poco sobre los efectos de esto**)

- Bottou recomienda: $f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right) \rightarrow \begin{cases} f(1) = f(-1) = 1 \\ \arg \max_x f''(x) = 1 \\ \text{ganancia cercana a 1} \end{cases}$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

74



BP: Normalizar las entradas

- Regla: la media de las entradas debe ser cero a lo largo de cada componente.
- Caso extremo: todos los valores positivos... implica que todos los pesos de la primera capa disminuyen o se incrementan al tiempo (gradientes con el mismo signo, ver Ec. De *update*)!
- Entonces: El vector de pesos tiene que cambiar de dirección zigzagueando - lento.
- Por la misma razón se prefieren sigmoideas simétricas... porque ellas generan la entrada para las demás capas.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

75



BP: Normalizar las entradas (2)

- Regla: La velocidad con la cual un peso cambia con descenso de gradiente es proporcional a la covarianza de la entrada.
- Para ecualizar las velocidades de aprendizaje las entradas se deben escalar para que tengan covarianzas iguales.
- Las entradas deben estar lo menos *correlacionadas* posible.
- Karhmen Loeve expansion (*KL-expansion*): Ilustrar en el **tablero**.
- En algunas aplicaciones esta transformación hace que se pierda información, p.e. pixeles y conexiones locales.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

76



BP: Inicialización de los pesos

- Regla: los pesos deben inicializarse de manera aleatoria, tratando que los productos internos queden en la parte no saturada de las funciones sigmoideas.
- Aleatoriedad: previene unidades aproximando la misma función (redundancia).
- Pesos grandes iniciales: saturan neuronas haciéndolas insensibles al aprendizaje (*flat spots*).
- Sugerencia1: para todas las componentes i de la neurona j :
$$w_i = 1/\sqrt{f_j} \quad \forall i \text{ con } f_j \text{ el número de entradas a la neurona } j$$
- Sugerencia2: Para tareas de clasificación, seleccionar como pesos valores normalizados aleatorios del *training set*.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

77



BP: Inicialización de la tasa de aprendizaje

- De nuevo: tasa grande implica velocidad pero inestabilidad. Tasa pequeña implica acercamiento *suave* al mínimo pero muy lento.
- Idea 1: igualar las velocidades de aprendizaje de diferentes unidades haciendo que la tasa de aprendizaje dependa del número de entradas. Así, la tasa para la neurona j sería:
$$\alpha_j = 1/f_j \text{ o } 1/\sqrt{f_j} \text{ con } f_j \text{ el fan-in}$$
- Idea 2: Le Cun *et al.* La rata óptima (a cada paso) es el inverso del valor propio más grande del *hessiano* de la función de error evaluada en w . MUY DEMORADO. Los mismos autores proponen un método analítico para aproximar este valor y otro iterativo.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

78



BP: Inicialización de la tasa de aprendizaje (2)

Inicialización de la tasa de aprendizaje (Cont.):

- Idea 3: Tasas adaptativas. Cambiar las tasas de acuerdo al cambio en el error.
 - Chan y Fallside, 1987: La tasa depende del ángulo entre el gradiente del error evaluado en dos iteraciones consecutivas.
 - Sutton, 1986: depende de cambios de signo en la derivada.
 - Franzisi, 1987: si el gradiente en pasos de tiempo consecutivo se *parece*, crece la tasa de aprendizaje. Si el gradiente cambia *considerablemente*, decrece la tasa.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

79



BP: Inicialización de la tasa de aprendizaje (2)

- Idea 4: para gradiente estocástico, decrecer la tasa:

$$\alpha^k = \frac{\alpha_0}{1+k}$$

Convergencia muy lenta... decrece muy rápido

$$\alpha^k = \frac{\alpha_0}{1+(k/\tau)}$$

Mejor, aunque implica seleccionar otro parámetro

$$\alpha^k = \alpha_0 \left(\frac{\alpha_f}{\alpha_0} \right)^{k/k_{\max}}$$

Se debe conocer k_{\max} .

- Intentar con una propia!

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

80



BP: Momentum

- Plaut *et al*, 1986.
- Suma un término que produce un efecto como de *inercia* en el movimiento sobre la superficie de error.

$$\Delta w_{ji}^k = -\alpha \frac{\partial J(w)}{\partial w_{ji}^k} + \rho \Delta w_{ji}^{k-1}, \text{ o escrito de manera recursiva:}$$

$$\Delta w_{ji}^k = -\alpha \sum_{n=0}^{N-1} \rho^n \frac{\partial J(w)}{\partial w_{ji}^{k-n}} + \rho^N \Delta w_{ji}^{k-N}$$

- El efecto del momento es similar al de las tasas adaptativas. Cuando la derivada es cercana a cero la tasa de aprendizaje se incrementa y cuando el gradiente fluctúa bastante el efecto del momento se desvanece.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

81



BP: Momentum (2)

Momentum, (Cont.).

- Si el error no cambia mucho la derivada es casi constante, además si N es grande:

$$\Delta w_{ji}^k \approx -\alpha \frac{\partial J(w)}{\partial w_{ji}^k} \sum_{n=0}^{N-1} \rho^n = -\frac{\alpha}{1-\rho} \frac{\partial J(w)}{\partial w_{ji}^k}$$

La tasa de aprendizaje α se ve aumentada por $1/(1-\rho)$

- Mientras que si el error cambia bastante, a largo plazo los sumandos de la expresión se *anulan*.

Momentum aumenta velocidad en *batch mode*, en modo incremental no es indispensable

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

82



BP: *Weight Decay y Pruning*

- Nunca se sabe cuantos parámetros debe tener una red para realizar bien la tarea.
- Una posible solución es entrenar con bastantes y luego aplicar *pruning algorithms*, Reed 1993.
- Decaimiento de pesos: a cada paso disminuir la magnitud de los pesos, de manera que caigan a cero si la conexión no se refuerza. Se puede lograr adicionando a la función objetivo un término que penalice pesos con alta magnitud:

$$J(w) = E(w) + \frac{\lambda}{2} \sum_i w_i^2 \Rightarrow \Delta w_i = -\alpha \frac{\partial E}{\partial w_i} - \alpha \lambda w_i$$

- Muchas otras funciones han sido propuestas. Al final, si un peso se hizo cero se *poda* la conexión. Si todos los pesos de una neurona se hacen cero, se elimina la unidad.

Feb-Jun, 2005

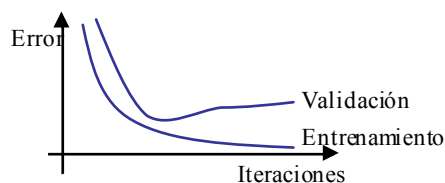
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

83



BP: *Cross-Validation*

- Separar la base de datos para: entrenamiento y para validación (A veces se sugiere dejar par *test* y reportar resultados sobre este *set*).
- Validación: resultado de la red ante entradas no presentadas en el entrenamiento... mide la capacidad de generalización.
- En redes entrenadas con backprop sobre **datos ruidosos**, el error de validación decrece hasta un mínimo y luego empieza a crecer aún cuando el error de entrenamiento siga disminuyendo.





Overfitting: red es sobre-entrenada se acomoda al ruido y pierde capacidad de generalización

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

84





Generalización

Chris Burgues:

the ability of the machine to learn any training set without error. A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree. Neither can generalize well. The exploration and formalization of these concepts has resulted in one of the shining peaks of the theory of statistical learning (Vapnik, 1979).

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 85



BP: Ejemplo

- EJEMPLO CON APROXIMACIÓN DE FUNCIONES PARA ILUSTRAR LA SUPERIORIDAD DE BACKPROP RESPECTO A POLINOMIOS PARA APROXIMAR FUNCIONES Y PARA MOSTRAR EL OVERFITTING CON DATOS RUIDOSOS (OBTENER GRAFICA DE ERROR DE TRAIN Y DE VALIDATION)

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 86



BP: Métodos de Segundo Orden

- Existen muchos métodos de segundo orden: incluyen información del hessiano para acelerar el aprendizaje.
- Emplean métodos indirectos de calcular la inversa del Hessiano.
- Ventaja: el hessiano es invariante ante transformaciones lineales de la entrada, e.g. escalamientos y rotaciones.
- La mayoría sólo aplican para *batch update*.
- Ejemplos: gradiente conjugado, Levenberg-Marquart, BFGS (Broyden, Fletcher, Goldfarb, Shanno).

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

87



BP: Variaciones para Aprendizaje Temporal

Variaciones importantes de *backprop* para aplicaciones que dependen del tiempo, e.g. control.

- *Time delay neural nets*
- Otras (no explicadas acá):
 - *Backprop through time*
 - *Time dependent recurrent backprop*
 - *Real-time recurrent learning*.

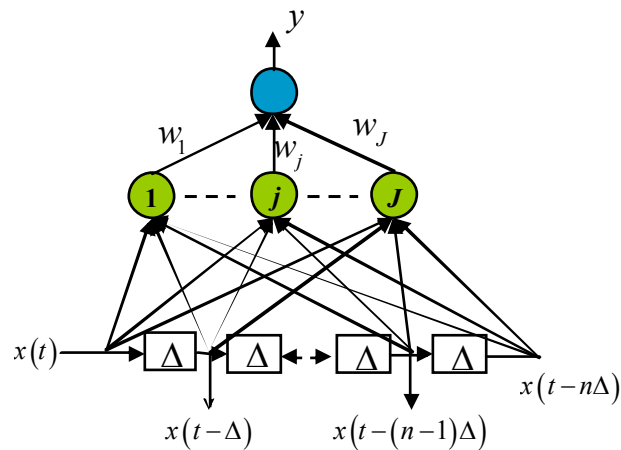
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

88



BP: *Time delay neural nets*



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

89



BP: *Time delay neural nets (2)*

- Puede ser utilizada para reconocer secuencias.
- Ha sido aplicada a reconocimiento de voz.
- Predicción: dada una secuencia de entrada:
 $\{x(t), x(t-1), \dots, x(t-(n-1)\Delta), x(t-n\Delta)\}$
predecir un valor futuro: $x(t+p)$
- El set de entrenamiento se construye con secuencias y “predicciones” conocidas.
- La salida de esta red es lineal y la función de la capa escondida es no lineal, p.e. sigmoideal.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

90



BP: *Time delay neural nets* (3)

- Se puede utilizar como reproductor de secuencias, si la salida predicha es: $x(t+p) = x(t+\Delta)$, y la salida es retrasada y realimentada a través de un *delay* a la red.
- ➔ Sólo funciona si la red es muy exacta.
- Existe una variación muy interesante para identificación y modelamiento de plantas. Consiste en entrar a la red una secuencia de entradas y la correspondiente secuencia de salidas entregadas por la planta. Entrenar dicha red hasta que reproduzca las mismas salidas.

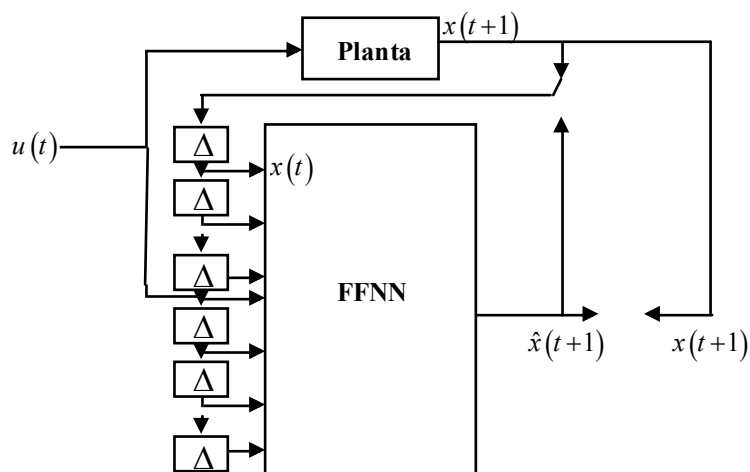
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

91




BP: *Time delay neural nets* (4)



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com


92



Contenido

- Introducción
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- **RBF: *radial basis function***
- Otras
- SVM: *support vector machines*

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 93



RBFN: *radial basis function networks*

- Redes RBF: redes de funciones base radiales.
- Redes neuronales *feed-forward*. Broomhead y Lowe 1988 entre otros.
- Motivadas por la observación de neuronas biológicas con respuesta local (Blanzieri, ENRICO).
- Arquitectura: primera capa de unidades con campos receptivos locales (o *kernels*) totalmente conectadas con una capa de salida de unidades lineales.
- El vector de entrada se presenta a todas las unidades de la primera capa.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com 94



RBFN (2)

- Los *kernels* definen el campo de recepción local.
- Buenas para aproximación y clasificación.
- Al igual que las FFNN con unidades sigmoidales, también son aproximadores universales.
- Para mi: son un tipo de SVM entrenadas de otra manera

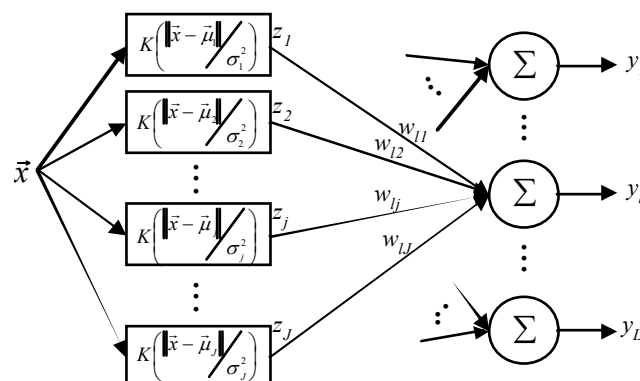
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

95



RBF: Arquitectura



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

96



RBF: *Kernel*

- Función estrictamente positiva con simetría radial
- Parametrizada por su “centro” (μ) y su “ancho” (σ).
- Alcanza su único valor máximo en el centro y decae, de manera controlada por σ , a medida que la distancia al centro aumenta

$$K(\vec{x}, \vec{\mu}, \sigma) > 0 \quad \forall \vec{x}$$

$$K(\vec{x}, \vec{\mu}, \sigma) = K(\vec{x}', \vec{\mu}, \sigma) \quad \forall \vec{x}' / \|\vec{x}' - \vec{\mu}\| = \|\vec{x} - \vec{\mu}\|$$

$$K(\vec{x}, \vec{\mu}, \sigma) < K(\vec{\mu}, \vec{\mu}, \sigma) \quad \forall \vec{x} \neq \vec{\mu}$$

$$K(\vec{x}, \vec{\mu}, \sigma) \geq K(\vec{x}', \vec{\mu}, \sigma) \quad \text{si} \quad \|\vec{x} - \vec{\mu}\| \leq \|\vec{x}' - \vec{\mu}\|$$

$$K(\vec{x}, \vec{\mu}, \sigma) \rightarrow 0 \quad \text{si} \quad \|\vec{x} - \vec{\mu}\| \rightarrow \infty$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

97



RBF: *Kernel* (2)

- Observe que realizan un mapeo:
 $R^n \rightarrow U \subset R^+$, donde generalmente $U = [0, 1]$
- Existen muchos tipos de *kenels* (más de esto en ANS)
- El más utilizado para redes RBF es el gaussiano:


$$K(\vec{x}, \vec{\mu}, \sigma) = K\left(\frac{\|\vec{x} - \vec{\mu}\|}{\sigma}\right) = \exp\left(-\frac{\|\vec{x} - \vec{\mu}\|^2}{2 \cdot \sigma^2}\right)$$

- Analizar el comportamiento del *kernel*.


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

98

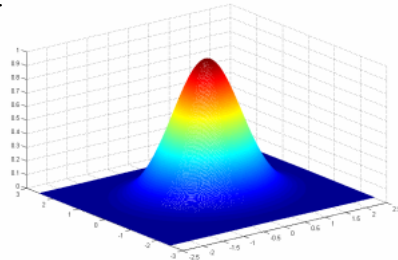


RBF: Ecuaciones



- La unidad j produce una salida:

$$z_j(\vec{x}) = K\left(\frac{\|\vec{x} - \vec{\mu}_j\|}{\sigma_j^2}\right)$$




- La salida de la neurona l de la capa de salida es:

$$y_l = \sum_{j=1}^J w_{lj} \cdot z_j(\vec{x})$$


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
 UPB. jeronimocm@yahoo.com

99



RBF: Entrenamiento



Que se debe entrenar?

- Centros de los *kernels*
- Anchos de los *kernel*. (Algunas veces se entrena la matriz de varianzas... Interesante!)
- Pesos de las unidades lineales de la capa de salida.

Numero de parámetros:

$$J \cdot L + L \cdot n + L \quad \text{o} \quad J \cdot L + L \cdot n + L \cdot n^2$$

Si se entrena matrices de varianzas

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
 UPB. jeronimocm@yahoo.com

100



RBF: Entrenamiento (2)

- Estrategia 1: es aplicar descenso de gradiente sobre los parámetros:

$$\Delta \mu_j = -\alpha_\mu \cdot \nabla_{\mu_j} J$$

$$\Delta \sigma_j = -\alpha_\sigma \cdot \nabla_{\sigma_j} J$$

$$\Delta w_{ij} = -\alpha_w \cdot \nabla_{w_{ij}} J$$

- Problema: Las superficies de error que genera las RBF poseen muchos mínimos locales. Además este tipo de entrenamiento es muy lento (tanto como backprop), debido principalmente a que no aprovecha la característica local de la red.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

101



RBF: Entrenamiento (3)

- Estrategia 2: Método de la pseudo-inversa. Consiste en fijar los centros aleatoriamente a patrones del *set*, fijar los anchos y luego despejar los pesos de la ecuación: $\vec{y} = W \cdot \vec{z}(\vec{x}) = \vec{d}$ calculando la pseudo-inversa de z se hallan los pesos.
- Estrategia 3: Método híbrido. Combina aprendizaje no supervisado y supervisado. Separa el aprendizaje de la capa escondida del de la capa de salida.
 - Entrena los centros mediante *clustering*
 - Calcula los anchos
 - Entrena la capa de salida con la regla delta.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

102



RBF: Entrenamiento Híbrido

Entrenamiento de los centros:

- Idea 1: dividir cada dimensión del espacio R^n en k particiones regulares formando una retícula de k^n hiper-paralelepípedos. Esta idea es impráctica para problemas multidimensionales.
- Idea 2: Técnica de *clustering*. P.e. *k-means* (más en ANS)
 - Se inicializan los centros c_i de manera aleatoria o tomando patrones.
 - Los centros son actualizados ante un patrón x_n según:

$$\mu_i = \begin{cases} \mu_i + \alpha_{\mu} \cdot (x_n - \mu_i) & \text{si } i = \arg \min_r \|x_n - \mu_r\| \\ c_i & \text{de otra manera} \end{cases}$$

A $i = \arg \min_r \|x_n - \mu_r\|$ se le llama unidad ganadora.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

103



RBF: Entrenamiento Híbrido (2)

Idea 2 (Cont.): Anchos

- Teóricamente si σ es igual para cada unidad, la red RBF es un aproximador universal.
- Una regla heurística para ajustar el ancho para todas las unidades escondidas, una vez ubicados los centros, es:

$$\sigma = \frac{\sum_{j=1}^J \|\mu_j - \mu_j^*\|}{J} \quad \text{donde } \mu_j^* \text{ es el centro más cercano a } \mu_j.$$

Otra posibilidad: ancho para cada unidad proporcional a la distancia al vecino más cercano:

$$\sigma_j = \rho \cdot \|\mu_j - \mu_j^*\|, \text{ normalmente } 1 < \rho < 1.5$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

104



RBF: Entrenamiento Híbrido (3)

Idea 2 (Cont.): Capa de Salida

- Después de haber entrenado la primera capa, los pesos de las unidades lineales de la capa de salida se ajustan con descenso de gradiente

$$\Delta w_{ij} = \alpha \cdot (d_l - y_l) \cdot \frac{\partial}{\partial w_{ij}} y_l = \alpha \cdot (d_l - y_l) \cdot \frac{\partial}{\partial w_{ij}} \left(\sum_{j=1}^J w_{ij} \cdot z_j(\vec{x}) \right)$$

$$\Delta w_{ij} = \alpha \cdot (d_l - y_l) \cdot z_j(\vec{x})$$

- Estrategia 4: EM, *expectation maximization*. Alto costo computacional pero con excelentes resultados (tal vez... mas en ANS).

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

105



RBF: Notas

- En muchos libros las redes RBF son consideradas aproximadores lineales... se debe a que se toma en cuenta solo la capa de salida.
- A la capa de entrada se le llama *extractor de características*. Lo que hace es transformar los puntos a un espacio de mayor dimensión donde éstos sean linealmente separables.
- A los pesos de cada unidad se les llama *prototipos* (*vector de soporte*).
- Muchos autores prefieren este tipo de redes por que: “es más fácil entender lo que pasa al interior de la red, una FFNN es una caja negra”.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

106



RBF: Notas (2)

- RBF puede alcanzar el desempeño de redes entrenadas con *backprop* en tiempos mucho menores, aunque requiere de un *set* de entrenamiento mayor.
- Existen métodos para sintonizar los campos receptivos mediante aprendizaje supervisado luego de haber aplicado *clustering*.
- Para difíciles tareas de clasificación RBF puede superar a *backprop*, sobre todo en menos *falsos positivos**. En *backprop* esto se puede reducir mediante: *training with rubbish*.
- *Backprop* es mejor extrapolando*.

Feb-Jun, 2005

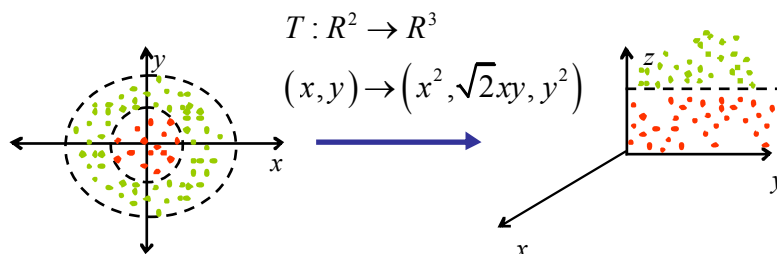
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

107



RBF: Ilustración


- Ilustración: transformación no lineal antes de aplicar un aproximador lineal.




Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

108




RBF: Ejemplo




- Ejemplo1: aproximación (mismo de *backprop*)
- Ejemplo2: clasificación (de pronto: ilustrar los falsos positivos)

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
109



RBF: Variaciones





- Normalización: Si la salida de la primera capa se normaliza, se mejoran las capacidades de aproximación (nuestra experiencia, más para regresión... para clasificación lo hace un poco lento... VER EJEMPLO).

$$z_j(\vec{x}) = K\left(\frac{\|\vec{x} - \vec{\mu}_j\|}{\sigma_j^2}\right) \cdot \left(\sum_{i=1}^J K\left(\frac{\|\vec{x} - \vec{\mu}_i\|}{\sigma_i^2}\right)\right)^{-1} \Rightarrow \sum_{j=1}^J z_j = 1$$

- Uso de *kernels* semilocales, p.e. *gaussian bar*: (ilustrar en tablero)

$$z_j(\vec{x}) = \sum_{i=1}^n w_{ji} \cdot \exp\left(-\frac{\|x_i - \mu_{ji}\|^2}{2 \cdot \sigma_{ji}^2}\right)$$



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
110

Contenido

- Introducción
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- Descenso de gradiente (*gradient descent*)
- μ -LMS y *Delta rule*
- Redes Multicapa: Algoritmo de *Backpropagation* para redes *feed-forward* (FFNN)
- Mejoras a *backprop*
- RBF: *radial basis function*
- **Otras**
- SVM: *support vector machines*

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com **111**

Otras

Existen muchos otros tipos de redes o de estructuras susceptibles de ser aprendidas:

- Sistemas difusos
- Clasificadores hiperesféricos, RCE.
- Y otros... no hay tiempo

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com **112**



RCE

- RCE, *Restricted Coulomb Energy*, Reilly 1982 y 1990.
- Es un clasificador hiperesférico, e.g. las superficies que separan el espacio no son hiperplanos sino hiperesferas.
- Una red adaptativa que ubica nuevas unidades (*unit allocation algorithms*).
- Arquitectura de dos capas: capa escondida de unidades hiperesféricas y capa de salida compuesta por **OR lógicas**.
- La entrada se conecta a todas las unidades de la capa escondida.
- La salida de cada unidad hiperesférica se conecta con sólo una OR de la salida.

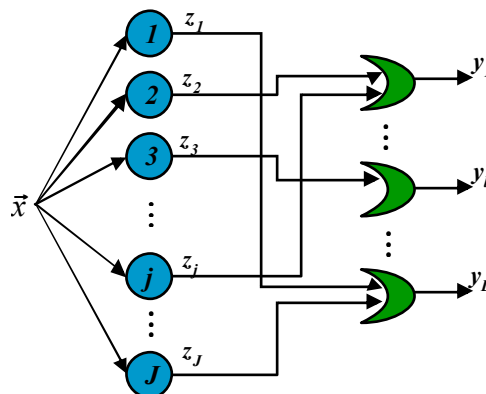
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

113



RCE: Arquitectura



Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

114



RCE: Funcionamiento

- Cada unidad en la capa de salida corresponde a una categoría.
- Si y_l esta activa, la entrada pertenece a la clase l . Si dos unidades se activan, se dice que la decisión es ambigua.
- La salida de la unidad j en la capa escondida es:
$$z_j(\vec{x}) = f[r_j - D(\mu_j, \vec{x})], \text{ con } \mu_j \in R^n \text{ es el centro, } r_j \in R \text{ es un umbral (radio), } D \text{ es una métrica y } f \text{ se define:}$$
$$f(a) = \begin{cases} 1 & a \geq 0 \\ 0 & \text{de otra manera} \end{cases}$$
- Cada que una entrada quede en la región de influencia de la unidad se activará la OR a la que esta está conectada.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

115





RCE: Entrenamiento

- Inicialmente: ninguna unidad.
- Se toma el primer patrón del set x^l , se ubica una unidad con centro en x^l y con radio r_{max} y se conecta con una OR en la salida.
- Se toma la segunda entrada x^2 , y:
 - Si x^2 dispara la unidad y pertenece a la clase de x^l , no se hace nada. En general si x^k dispara más de una unidad, se deben disminuir los radios de las regiones a las que x^k no pertenezca.
 - Si x^2 pertenece a la categoría de x^l , pero no dispara la unidad, se crea una nueva unidad con centro en x^2 y radio r_{max} . En general, si x^k no dispara ninguna unidad se crea una nueva con centro en x^k y radio $r = \min(r_{max}, d_{min})$, donde d_{min} es la distancia desde x^k al centro más cercano de otra categoría. La salida de esta unidad se conecta con la OR de la clase.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com



116

RCE: Entrenamiento (2)

- Continuación...
 - Si x^2 no pertenece a ninguna categoría representada por la red. Se ubica una nueva unidad como en el primer paso con el $r = \min(r_{max}, d_{min})$. Se adiciona también una OR representando esta nueva categoría. Si otras unidades se activan, sus radios se decrecen.
- El algoritmo se detiene cuando no se creen más unidades y los radios no sufran variaciones importantes.



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
117

RCE: Notas

- Es capaz de resolver problemas no linealmente separables.
- Puede generar regiones disyuntas con facilidad.
- Puede funcionar dinámicamente, e.g. se pueden añadir nuevas categorías si necesidad de *reaprender* lo anterior.
- Sólo para clasificación.
- Por lo general utiliza más unidades de las necesarias.
- Algunos métodos mas refinados incluyen crecimiento de radios.

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
118





RCE: Ejemplo

- EJEMPLO DE 4 CATEGORÍAS.

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

119



Contenido

- Introducción
- *Perceptron learning Rule*
- *Adaline (α -LMS)*
- *Descenso de gradiente (gradient descent)*
- μ -LMS y *Delta rule*
- *Redes Multicapa: Algoritmo de Backpropagation para redes feed-forward (FFNN)*
- *Mejoras a backprop*
- *RBF: radial basis function*
- *Otras*
- **SVM: support vector machines**

Feb-Jun, 2005 AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

120



SVM: *support vector machines*

- Arquitectura de una sola capa escondida de unidades no lineales y una capa de salida con:
 - Función signo para **clasificación**.
 - Función identidad (e.g. salida lineal) para **regresión**.
- Nace del análisis estadístico y se basa en el principio de *structural risk minimization*: buscar el mínimo sobre todas las posibles funciones implementables por una máquina.
- *Statistical learning theory*: Vapnik y Chervonenkis, 60s.
- También: *kernel learning*, *kernel machines*.
- Principales: Bernhard Schölkopf, Alexander J. Smola (ANU) y otros. www.kernel-machines.org

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

121



SVM (2)

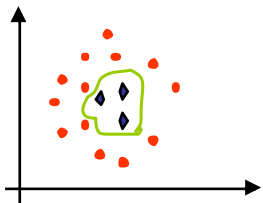
- Datos: patrones generados por una *regularidad* estocástica desconocida ($P(x,y)$).
- Aprendizaje: extracción de la regularidad a partir de los datos.
- *Capacidad*: tipos de funciones que una *máquina* puede generar.
- *Support vector machines*: utilizan funciones de clasificación en un espacio diferente del de entrada llamado *feature space* (normalmente de más dimensiones).
- Hallar el hiperplano óptimo (similar al *perceptron*) que separe los puntos en el *espacio de características*.

Feb-Jun, 2005


AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

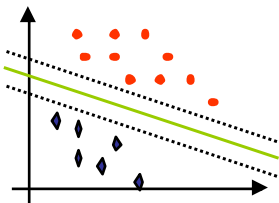
122

SVM (3)



Espacio de entrada

$\varphi: R^2 \rightarrow R^h,$
 $2 \ll h$




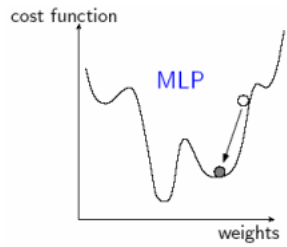
Feature space

Kernel trick: no se necesita conocer el espacio generado por φ explícitamente!!

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
123

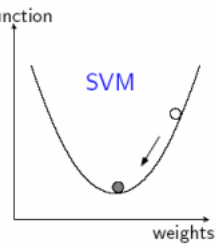
SVM (4)

- Una de las ventajas de los SVM sobre las *Multi Layer Perceptron* es que la función objetivo es convexa y por lo tanto tiene un solo mínimo.



MLP


→




SVM

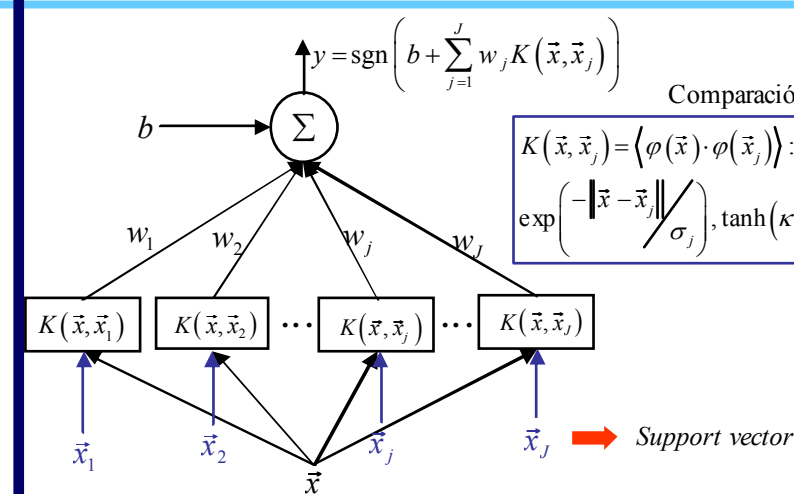
Tomado de: Support Vector Machines and Kernelbased Learning. Johan Suykens. K.U. Leuven, ESAT-SCD-SISTA. <http://www.esat.kuleuven.ac.be/sista/members/suykens.html>

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
124





SVM: Arquitectura





Comparación:

$$K(\vec{x}, \vec{x}_j) = \langle \varphi(\vec{x}) \cdot \varphi(\vec{x}_j) \rangle : (\vec{x} \cdot \vec{x}_j + c)^d,$$

$$\exp\left(-\frac{\|\vec{x} - \vec{x}_j\|}{\sigma_j}\right), \tanh(\kappa(\vec{x} \cdot \vec{x}_j) + \theta)$$

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
125





SVM: Explicación

- Como de costumbre: problema de optimización.
- Se resuelve aplicando *multiplicadores de Lagrange*.
- Para el **caso lineal**: Objetivo, minimizar el Lagrangiano:

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^N \alpha_i [d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1]$$

Con N el número de datos de entrenamiento, d_i es la salida deseada ante x_i y $\alpha_i \geq 0$ los multiplicadores de Lagrange.

- A encontrar: \vec{w} y b . Los multiplicadores de Lagrange son una herramienta, no se necesitan al final.
- Observe que si el dato esta bien clasificado: $d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1$

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
126



SVM: Explicación (2)

- El uno de la expresión representa el *margin*. La variación *C-SVM* utiliza un *soft-margin*.
 - Importante: SVM busca encontrar el hiperplano con el mayor margen, sujeto a unas restricciones... por eso se utilizan los multiplicadores de Lagrange.
- “The hyperplane that maximizes the minimum distance from the hyperplane to the closest training point”**
- Entre más pequeño sea el vector de pesos que describe el plano más grandes es el margen,

(ver próxima)

$$\text{margin} \sim 1/\|\vec{w}\|$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

Feature space 127



SVM: Explicación (3)

- El hiperplano con mayor margen es aquel para el cual la norma de w es mínima:

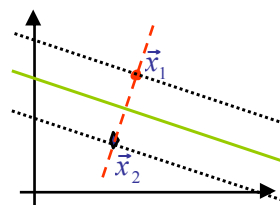
$$\langle \vec{w}, \vec{x}_1 \rangle + b = +1, \langle \vec{w}, \vec{x}_2 \rangle + b = -1,$$

restando y por la bilinealidad del producto interno:

$$\langle \vec{w}, \vec{x}_1 - \vec{x}_2 \rangle = \|\vec{w}\| \cdot \|\vec{x}_1 - \vec{x}_2\| \cos(\xi) = 2$$

$$\Rightarrow \|\vec{x}_1 - \vec{x}_2\| = 2/\|\vec{w}\| \cos(\xi)$$


$$\vec{x}_1 \parallel \vec{x}_2 \parallel \vec{w} \Rightarrow \text{margin} \equiv 1/\|\vec{w}\|$$




Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

128



SVM: Explicación (4)




- Problema de optimización:


$$\begin{cases} L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^N \alpha_i [d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1] \\ d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) > 1 \end{cases}, \alpha_i \geq 0 \forall i$$
- Sistema de ecuaciones... inecuaciones

$$\begin{cases} \nabla L(\vec{w}, b, \vec{\alpha}) = \vec{0} \\ d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) > 1, \alpha_i \geq 0 \forall i \end{cases}$$

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
129



SVM: Explicación (5)



- Iguando el gradiente a cero:

$$\frac{\partial}{\partial b} L(\vec{w}, b, \vec{\alpha}) = \frac{\partial}{\partial b} \left(\frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^N \alpha_i [d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1] \right) = 0 \Rightarrow \sum_{i=1}^N \alpha_i d_i = 0$$

$$\frac{\partial}{\partial \vec{w}} L(\vec{w}, b, \vec{\alpha}) = \frac{\partial}{\partial \vec{w}} \left(\frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^N \alpha_i [d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1] \right) = 0 \Rightarrow \vec{w} = \sum_{i=1}^N \alpha_i d_i \cdot \vec{x}_i$$
- Para minimizar L :

Si $d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1 < 0$, α_i crecerá y \vec{w}, b deberán cambiar para minimizar L

Si $d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) - 1 > 0$, α_i decrecerá hasta cero.

➔ **Son las KKT Conditions... Kuhn Tucker**

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
130



SVM: Explicación (4)

- Así:

$$d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) > 1 \Rightarrow \alpha_i = 0, \vec{x}_i \text{ irrelevante}$$

$$d_i \cdot (\langle \vec{w}, \vec{x}_i \rangle + b) = 1 \Rightarrow \text{en el margen, } \vec{x}_i \text{ "Support Vector"}$$

- La salida depende de los patrones en el margen:

$$f(\vec{x}) = \text{sgn}(b + \langle \vec{w}, \vec{x} \rangle), \text{ por la bilinealidad,}$$

$$f(\vec{x}) = \text{sgn}\left(b + \left\langle \sum_{j=1}^N \alpha_j d_j \vec{x}_j, \vec{x} \right\rangle\right) = \text{sgn}\left(b + \sum_{j=1}^N \alpha_j d_j \langle \vec{x}_j, \vec{x} \rangle\right)$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

131



SVM: Explicación (5)

- Dual Problem*: consiste en reescribir la función objetivo en términos del producto interno:

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \left\| \sum_{j=1}^N \alpha_j d_j \vec{x}_j \right\|^2 - \sum_{i=1}^N \alpha_i \left[d_i \cdot \left(\left\langle \sum_{j=1}^N \alpha_j d_j \vec{x}_j, \vec{x}_i \right\rangle + b \right) - 1 \right]$$

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \left\| \sum_{j=1}^N \alpha_j d_j \vec{x}_j \right\|^2 - \sum_{i=1}^N \sum_{j=1}^N \alpha_j \alpha_i d_i d_j \langle \vec{x}_j, \vec{x}_i \rangle - \sum_{i=1}^N \alpha_i (d_i b - 1)$$

$$\text{Pero: } \sum_{i=1}^N \alpha_i d_i = 0$$

$$L(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^N \alpha_i + \frac{1}{2} \left\| \sum_{j=1}^N \alpha_j d_j \vec{x}_j \right\|^2 - \sum_{i=1}^N \sum_{j=1}^N \alpha_j \alpha_i d_i d_j \langle \vec{x}_j, \vec{x}_i \rangle$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

132



SVM: Explicación (6)

- Se llega a que se debe maximizar:

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_j \alpha_i d_i d_j \langle \vec{x}_j, \vec{x}_i \rangle, \text{ sujeto a}$$

$$\sum_{i=1}^N \alpha_i d_i = 0, \alpha_i \geq 0 \forall i$$

Quadratic optimization problem

- Se debe buscar los α_i que maximicen esta función. Luego se despejan los pesos y el parámetro b .

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

133



SVM: Explicación (7)

- Queda entonces un problema que solo depende de productos internos... se puede aplicar el *truco del kernel* (Mercer Theorem):

$$\langle \varphi(\vec{x}_j), \varphi(\vec{x}_i) \rangle = k(\vec{x}_j, \vec{x}_i)$$

Permite resolver problemas no linealmente separables

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_j \alpha_i d_i d_j k(\vec{x}_j, \vec{x}_i), \text{ sujeto a}$$

$$\sum_{i=1}^N \alpha_i d_i = 0, \alpha_i \geq 0 \forall i$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

134



SVM: Explicación (8)

- Ejemplito:

$$\varphi: R^2 \rightarrow R^3, \varphi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\langle \varphi(\vec{x}), \varphi(\vec{x}') \rangle = \left\langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (x_1'^2, \sqrt{2}x_1'x_2', x_2'^2) \right\rangle$$

$$\langle \varphi(\vec{x}), \varphi(\vec{x}') \rangle = x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 = (x_1x_1' + x_2x_2')^2$$

$$\langle \varphi(\vec{x}), \varphi(\vec{x}') \rangle = \langle \vec{x}, \vec{x}' \rangle^2 = k(\vec{x}, \vec{x}')$$

→ El producto interno en el espacio de características se puede escribir en términos de productos internos en el espacio original

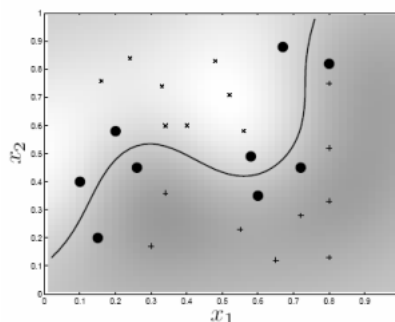
Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

135



SVM: Ilustración





Tomado de: Support Vector Machines and Kernelbased Learning. Johan Suykens. K.U. Leuven, ESAT-SCD-SISTA. <http://www.esat.kuleuven.ac.be/sista/members/suykens.html>

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

136






SVM: Entrenamiento

- Utilizar el método que más le guste para solucionar el COP (*constrained optimization problem*)!...
- Batch: resolviendo todo de una vez... costosísimo $O(N^3)$
- Ascenso de gradiente: lento
- Varios métodos en la literatura: GDA, SMO...

Problema: (mio)... como encontrar los SV's. Una vez se tengan los SV's la solución es sencilla

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
137

SVM: Entrenamiento (2)

GDA, *General decomposition algorithm*, Osuna et al., 1997:

- Descomponer el problema: escogiendo vectores al azar (tanto como se necesite para que el sistema sea conformable??)
- Revisar criterios de convergencia, es decir, que clasifique todos bien.
- Escoger otro posible *set* de SV's y volver a iterar.

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
138



SVM: Entrenamiento (3)

SMO, *Sequential Minimal Optimization*, Platt 1999 en Schölkopf, Burgues and Smola editors.

- Optimiza sólo dos α_i en cada iteración. Estos valores se seleccionan con una heurística.
- El problema resultante se soluciona analíticamente.
- Se itera hasta satisfacer algún criterio de paro.

Rápido, no necesita almacenar la matriz de kernel en memoria (los otros lo hacen).

$$\{k(\vec{x}_j, \vec{x}_i)\}_{N \times N}$$

Feb-Jun, 2005

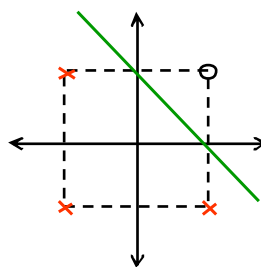
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

139



SVM: Ejemplo Lineal

- Tablero: AND



$$\text{Soln: } y = -x + 1$$

Observe:

$$d(\vec{x}^1, \text{recta}) = d(\vec{x}^2, \text{recta}) = d(\vec{x}^3, \text{recta}) = \frac{\sqrt{2}}{2}$$

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

140



SVM: Ejemplo no Lineal

- **XOR**

$$K(\vec{x}, \vec{x}_i) = (\vec{x}^T \cdot \vec{x}_i + 1)^2$$

$$= 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2}$$

$$K(\vec{x}, \vec{x}_i) = \langle \varphi(\vec{x}), \varphi(\vec{x}_i) \rangle$$

$$\varphi(\vec{x}) = [1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]^T$$

El *kernel* escogido realiza una transformación a un espacio de 6 dimensiones... aunque no se necesita conocerla!

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

141



SVM: Ejemplo no Lineal (2)

- La función a optimizar queda:

$$W(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

- Los valores óptimos son: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$


- Pesos: $w = \sum_{i=1}^N \alpha_i d_i \varphi(x_i) = [0, 0, -1/\sqrt{2}, 0, 0, 0]$

- Salida: $y \sim \vec{w}^T \varphi(\vec{x}) = -x_1 \cdot x_2$


Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

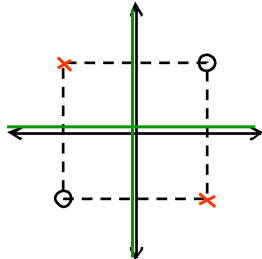
142




SVM: Ejemplo no Lineal (3)




- Hyperplano óptimo:
 $-x_1 \cdot x_2 = 0$



Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
143



SVM: Variaciones



- Para no exigir que el problema sea linealmente separable se introduce *soft margin*, se conoce como C-SVM.

$$J(\vec{w}, \vec{\zeta}) = \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^N \zeta_i$$

- ν-SVM

Feb-Jun, 2005
AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com
144



- Spike sims.

Feb-Jun, 2005

AS. Diplomado en Computación Inteligente.
UPB. jeronimocm@yahoo.com

145