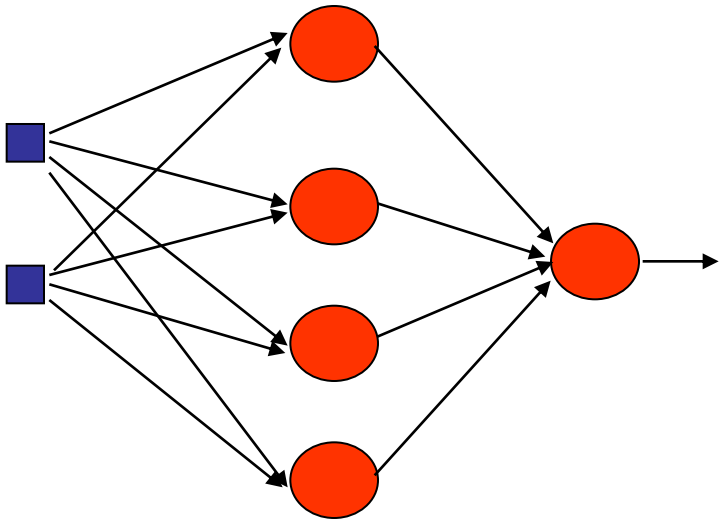
Redes Neuronales con Base Radial (Poggio y Girosi,1990)



Esquema de trabajo:

- Estructura de una RBF
- Problema de clasificación (otra vez)
  - Ejemplo XOR
  - Teorema de Cover (número de neuronas)
- Problema de Interpolación (aproximación) de funciones
  - Aproximadores universales
  - RBF
  - RRBF
  - GRBF
  - Entrenamiento
- Comparación MLP vs RBF
- Resumen
- Practica

Existe una clase de técnicas en las cuales los puntos de entrenamientos son utilizados no solo en la fase de entrenamiento sino también en la predicción. Algunos caen dentro de la familia de métodos denominados "basados en memoria" .

Típicamente estos métodos requieren de alguna forma o métrica de distancia entre puntos para medir similaridades. Estos métodos suelen ser rápidos en el entrenamiento pero lentos en las predicciones.

Muchos modelos pueden ser reformulados en su representación dual en las que las predicciones son basadas en combinaciones lineales de funciones kernel evaluados en los puntos de entrenamiento

Para modelos que están basados en un mapa de *feature space* $\varphi(x)$, la función de kernel está dada por

$$k(x, x') = \varphi(x)^t \varphi(x')$$

De esta definición se desprende que la función kernel es simétrica ($k(x, x') = k(x', x)$).

El concepto del kernel formulado como un producto interno del espacio de rasgos permite hacer extensiones interesantes de algoritmos conocidos usando el *truco del kernel,* que consiste en sustituir los lugares donde aparezca el producto interno de vectores (representación dual).

Se pueden construir kernels directamente. Suponga $k(x,z) = (x^t z)^2$.
Y un espacio en dos dimensiones, de modos que $x = (x_1, x_2)$, **podemos expandir e identificar el mapa no-lineal :**

$$\begin{aligned} k(x,z) &= (x^t z)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1{}^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)\left(z_1^2, \sqrt{2}z_1 z_2, z_2^2\right)^t \end{aligned}$$

$\varphi(x)=(x_1^2, \sqrt{2}x_1 x_2, x_2^2)$

Se pueden construir kernels a partir de otros…..

$$
\begin{aligned}
k(x, x') &= c k_1(x, x') \\
k(x, x') &= f(x) k_1(x, x') f(x') \\
k(x, x') &= q(k_1(x, x')) \\
k(x, x') &= exp(k_1(x, x')) \\
k(x, x') &= k_1(x, x') + k_2(x, x') \\
k(x, x') &= k_3(\varphi(x), \varphi(x')) \\
k(x, x') &= k_1(x, x') k_2(x, x')
\end{aligned}
$$

c >0 una constante, $f(.)$ una función , $q(.)$ un polinomio con constantes no negativas, φ una función de x a R^M

Ejemplo: $k(x, x') = exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ **es un kernel**

$$\|x - x'\|^2 = x^t x + (x')^t x' - 2x^t x'$$

Esto arroja:

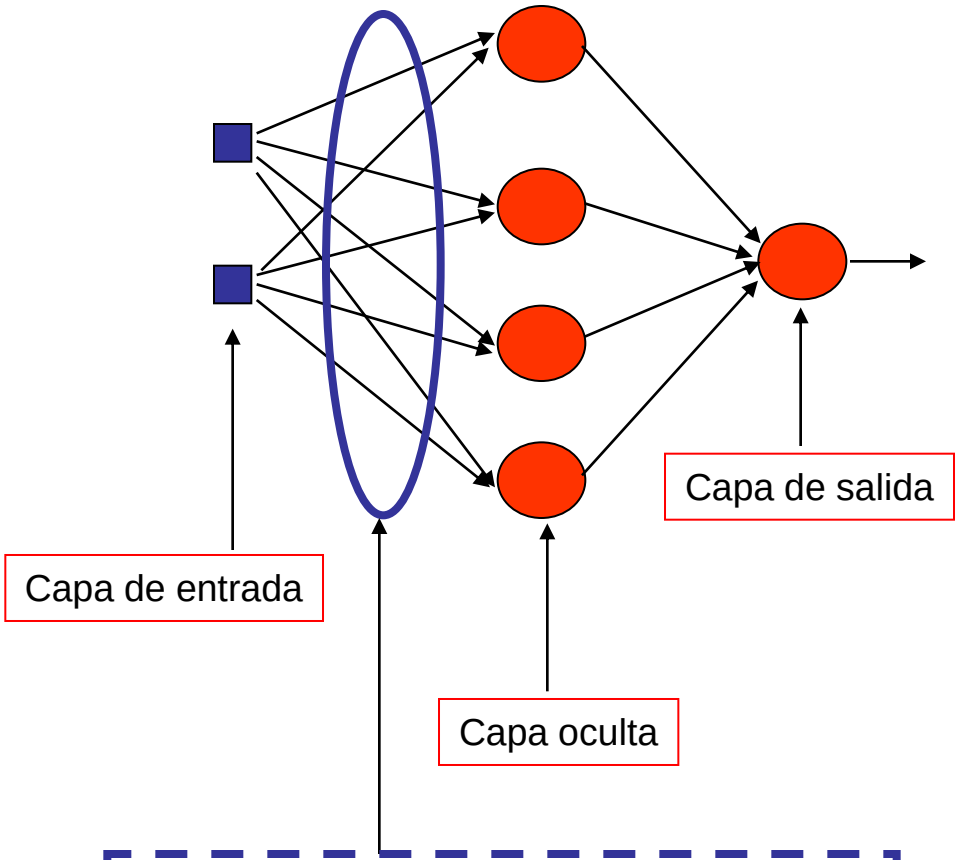
$$k(x, x') = exp\left(-\frac{x^t x}{2\sigma^2}\right) \exp\left(\frac{x^t x'}{\sigma^2}\right) \exp\left(-\frac{(x')^t x'}{2\sigma^2}\right)$$

Estructuralmente una RFB es como una MLP.

Porqué la MLP puede resolver problemas de separabilidad lineal? Existe una capa intermedia que no es otra cosa que la transformación de los datos de entrada a otro espacio y en la última capa se puede hacer linealmente separable.

Nos interesan los problemas de clasificación nolineal (los lineales los resolvemos muy fácilmente!)

Estructura de la red:



Estos pesos tienen la particularidad que no se calculan, ellos están determinados por la función **distancia**

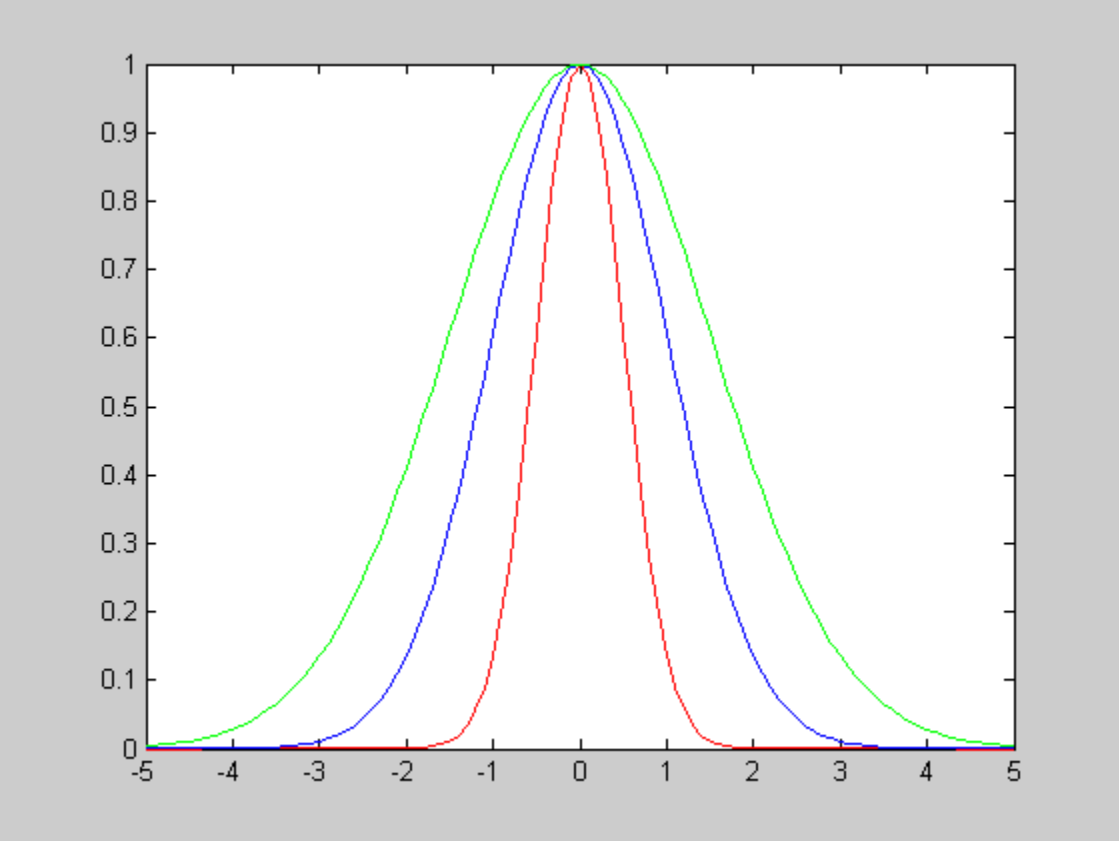
Capa de entrada: tiene tantos dispositivos como dimensionalidad de la data.

Capa oculta: Hay una sola, neuronas no-lineales usualmente con funciones de activación radial. El número de neuronas depende de la aplicación. Todas las neuronas son distintas. Usualmente muchas neuronas.

Capa salida: neurona lineal. Los pesos entre la capa oculta y esta deben ser determinados

Funciones de Activacion capa oculta:

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{con} \quad r \in \Re$$



(σ=0.5,1.0,1.5)

Problema de clasificación

Sea H el conjunto de patrones de entrada que pertenecen a una clase. $H = H_1 U H_2 = \{x_1, x_2, \dots, x_N\}$

Se dice que el conjunto es φ **−separable si para un conjunto de funciones** $\{\varphi_1, \varphi_2.., \varphi_{m1}\}$, **existe un vector** $W \in Rm^1$ **tal que**

$$W^t\varphi(x) > 0, \quad x \in H_1$$
$$W^t\varphi(x) < 0, \quad x \in H_2$$

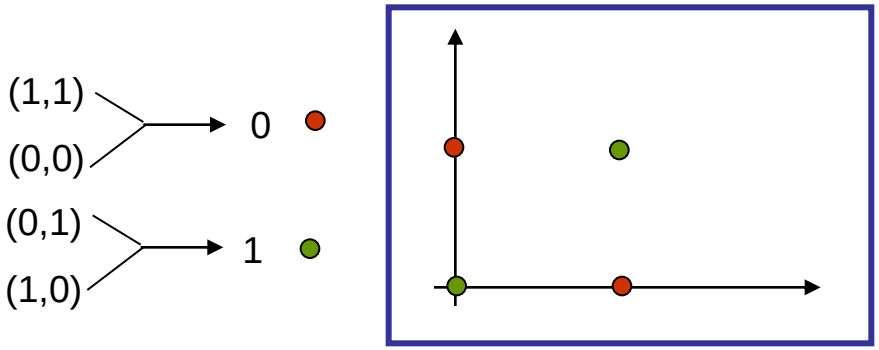
Las funciones φ **serán en un futuro las funciones de capa oculta**

$W^t\varphi(x) = 0$ **es la superficie de separación (ya no es un hiperplano).**

Donde $\varphi(x) = [\varphi_1, \varphi_2, \cdots, \varphi_{m1}(x)\ ]^t$

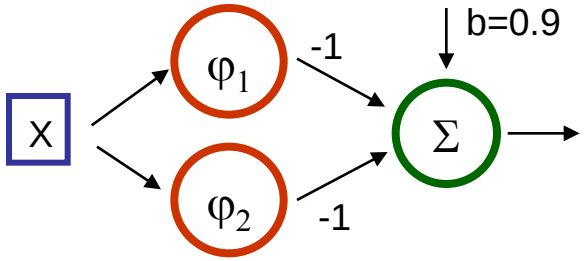
En esencia estamos transformando un problema de separabilidad no-lineal a uno lineal!

Ejemplo:XOR

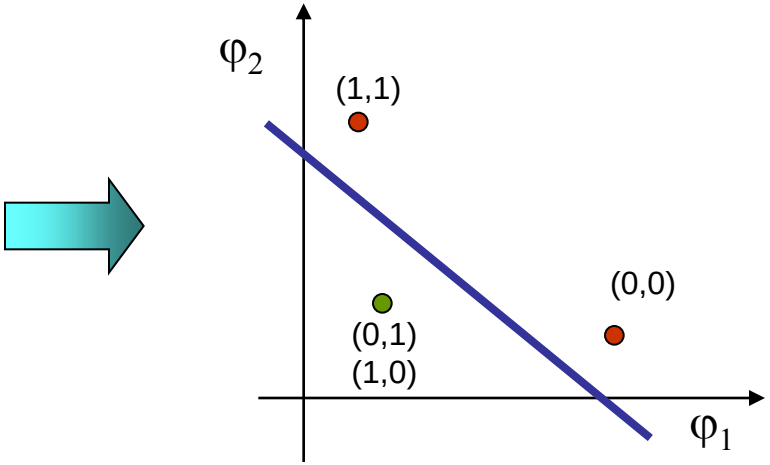


Sean $\varphi_1(x) = \exp(-\|x-x_1\|)$

$\varphi_2(x) = \exp(-\|x-x_2\|)$



	φ_1	φ_2
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678



El teorema de Cover dice que un problema de clasificación de patrones es más probable que sea lenealmente separable en un espacio de dimensión mayor. Mientras mayor sea la dimesión del espacio, mayores serán tus posibilidades de separar linealmente.

En la estructura de una RBF está implícita esta transformación a un espacio de dimensión m_1. (El espacio de entrada tiene dimensión m_0). Mientras más alto sea m_1 mejor son las posibilidades de éxito en la clasificación lineal a posteriori.

Estas funciones de la capa oculta generan un espacio que pueden ser aproximadas por variedades de orden r.

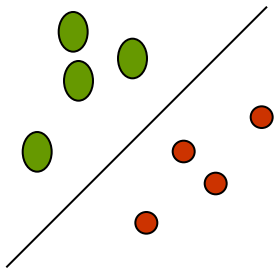
Cuantas Neuronas hacen falta en la capa oculta?

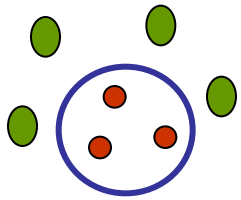
El teorema de Cover nos dice que mientras mas neuronas coloquemos en la capa oculta la probabilidad de que una dicotomía (particion binaria) sea φ – separable aumenta.

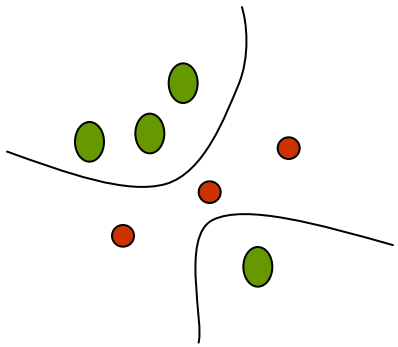
$$\sum_{0 \le i_1 \le i_2 \le \cdots \le i_r \le m_0} a_{i_1 i_2 \cdots i_r} x_{i_1} x_{i_2} \cdots x_{i_r} = 0$$

$\varphi(x) = x_{i1} x_{i2} \ldots . x_{ir}$ son funciones no lineales (variedad racional de orden r)

r=1 → hiper-planos;
r=2 → hiper-parábolas
r=2 + cond. En coeficientes → hiper-esferas







Cuantos patrones son separables con m_1 neuronas?

Si $X_1, X_2, \ldots, X_N$ es una secuencia de patrones (indep). Sea N (var. aleatoria) el mayor entero tal que la secuencia es φ - separable

$$P(N=n) = \left(\frac{1}{2}\right)^n \binom{n-1}{m_1-1}$$

E(N)=2m_1

OJO:

Aunque las formas de la dem. No son iguales al de las funciones radiales, el argumento puede mantenerse.

Con el problema del O exclusivo bastó con dos neuronas.

Cuantas Neuronas hacen falta en la capa oculta?

Muchas!

Asintóticamente N=2m_1
(si tengo m_1 neuronas puedo esperar separar hasta 2m_1 patrones)

Problema de Interpolación

- Es una técnica para hacer ajustes de funciones o superficies
- Utiliza los datos para hacer tal aproximación
- Interpola los datos restantes (nuevos)

Queremos encontrar para N puntos $\{X_i \in \Re^{m_0}\}$

distintos con salida deseada $\{d_i \in \Re\}$

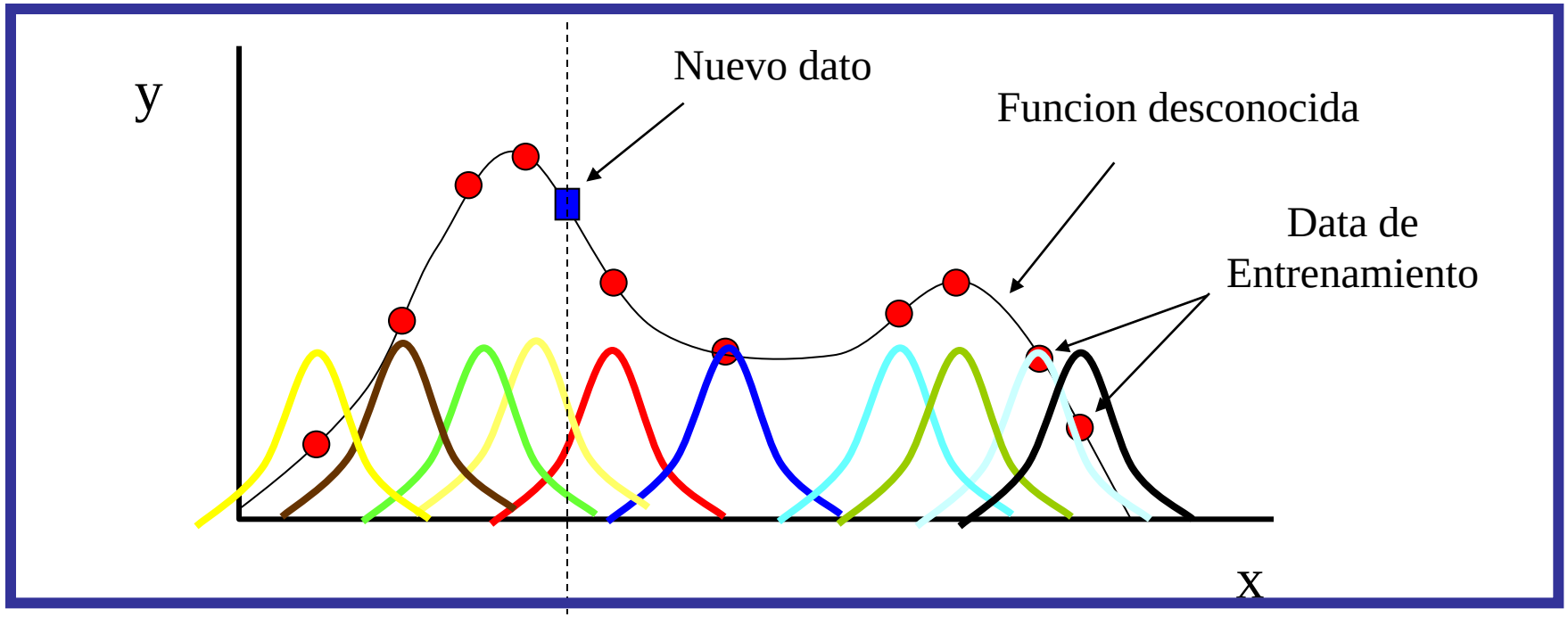
Una función $F : \Re^N \rightarrow \Re^1$ tal que

$$F(X_i) = d_i \quad i = 1,2,....,N$$

Un aproximador universal es buscar F de la forma

$$F(X) = \sum_{i=1}^{N} w_i \phi\left(\|X - X_i\|\right) \quad i = 1,2,3...,N$$

$$\left\{\phi\left(\|X - X_i\|\right) \mid \quad i = 1,2,3...,N\right\}$$ Funciones no-lineales arbitrarias



Los X_i son los centros, por esto decimos que todas las neuronas son distintas, que es una diferencia importante de los perceptrones multicapas visto anteriormente.

$$F(X_i) = d_i \quad i = 1,2,....,N$$

$$+$$

$$F(X) = \sum_{i=1}^{N} w_i \phi\left(\|X - X_i\|\right)$$

$$\Rightarrow \quad \begin{aligned} w_1\phi_{11} + \quad & w_2\phi_{12} + \quad \cdots \quad + w_N\phi_{1N} = d_1 \\ w_1\phi_{21} + \quad & w_2\phi_{22} + \quad \cdots \quad + w_N\phi_{2N} = d_2 \\ \vdots \quad & \vdots \quad \cdots \quad \vdots \\ w_1\phi_{N1} + \quad & w_2\phi_{N2} + \quad \cdots \quad + w_N\phi_{NN} = d_N \end{aligned}$$

Con $\quad \phi_{ji} = \phi\left(\|x_j - x_i\|\right)$

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & & & \phi_{2N} \\ \vdots & & & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix}$$

Matriz de interpolación

$$\mathbf{\Phi w = d}$$

ó

$$\boldsymbol{w = \Phi^{-1} d}$$

Realmente puedo hacer esto?

Es invertible la matriz?

Teorema (Micchelli, 1986): **La matriz Φ es nosingular si todos los valores de la data son distintos y las funciones φ son de un cierta clase amplia (que incluyen las multicuadráticas, multicuádraticas inversas y las gausianas)**

- Multicuadrática

$$\phi(r) = \left(r^2 + c^2\right)^{1/2} \quad c > 0 \quad \text{con} \quad r \in \Re$$

- Multicuadrática Inversa

$$\phi(r) = \frac{1}{\left(r^2 + c^2\right)^{1/2}} \quad c > 0 \quad \text{con} \quad r \in \Re$$

- Gausiana

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{con} \quad r \in \Re$$

Resumen

- **Ajusta el 100% de la data de entrenamiento (error=0)**
- **Entrenamiento (en dos etapas) no iterativo.**
  - **Primera etapa: determinar las neuronas (usando unicamente los datos).**
  - **Segunda etapa:Determinar los pesos resolviendo el sistema de ecuaciones.**
- **Qué ocurre con la generalización?**

En la práctica podemos tener sistemas sobre determinados. El sistema podría ajustar data con ruido (agrega incertidumbre).

La generalización está afectada por:

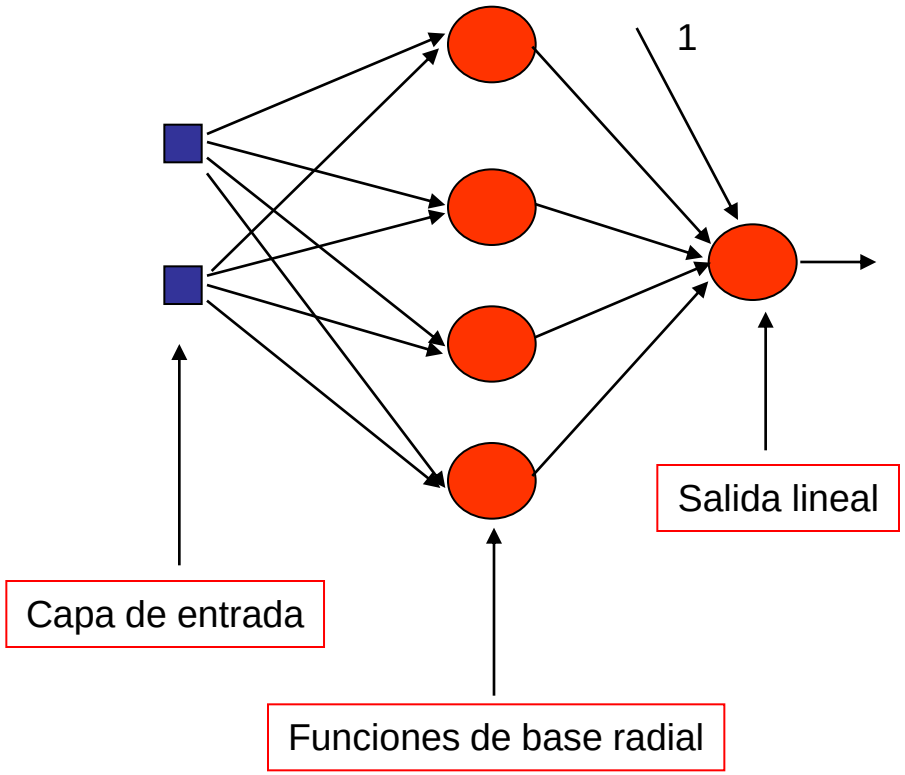
1. **Datos (tamaño de la muestra):** $N = O(w/\epsilon)$
2. **Arquitectura de la red**
3. **Complejidad del problema (sin control sobre esto!!)**

Si entrenamos con una data que contiene ruido, la función que interpola exactamente es altamente oscilatoria (poco deseable). Queremos una función mas suave.

Modificaciones:

1. El número de funciones no es exactamente el número de datos.
2. Los centros no necesariamente son los mismos datos.
3. Los parametros de las neuronas (σ) no tienen que ser los mismos para cada neurona.
4. Incluimos el sesgo (compensa con el valor promedio de las funciones de activación y los correspondientes valores deseados).

Ahora tenemos una RBF (Radial Basis Function) o Red de Base Radial



$$y(x) = \sum w_j \varphi_j(x) + w_0 = \sum w_j \varphi_j(x) \qquad \text{Con } j = o,,,,M < N$$

$$\varphi_j(x) = \exp(-\|x - t_j\|^2 / 2\sigma^2)$$

Tenemos un sistema de la forma $Y = W\Phi$

Podemos encontrar los pesos, mediante la minimización de la función de error cuadrático. Con las neuronas fijas (los centros y el parámetro de dispersión), podemos buscar minimizar la función de error con respecto a los pesos. Esto resulta en resolver la ecuación: (ver tarea)

$$\Phi^t \Phi \mathbf{w}^{\mathbf{t}} = \Phi^t D \quad (1)$$

La solución involucra la pseudoinversa de $\Phi = [\,(\,\Phi^T\Phi\,)^{-1}\,\Phi^T\,]$.

En la práctica el sistema (1) se resuleve usando descomposición en valores singulares.

Otra forma de garantizar regularización (usando todos los datos) es...

$$\min E(F) = \frac{1}{2}\sum_{i=1}^{N}\left[d_i - F(x_i)\right]^2 + \frac{1}{2}\lambda\|DF\|^2$$

Error de aproximación

Penalización

D es un operador lineal diferencial,
λ es un parámetro de regularización

La solución es de la forma:

$$F_\lambda(x) = \frac{1}{\lambda}\sum_{i=1}^{N}\left[d_i - F(x_i)\right] * G(x, x_i)$$

$$F_\lambda(x) = \frac{1}{\lambda}\sum_{i=1}^{N}\left[d_i - F\left(x_i\right)\right] * G(x, x_i)$$

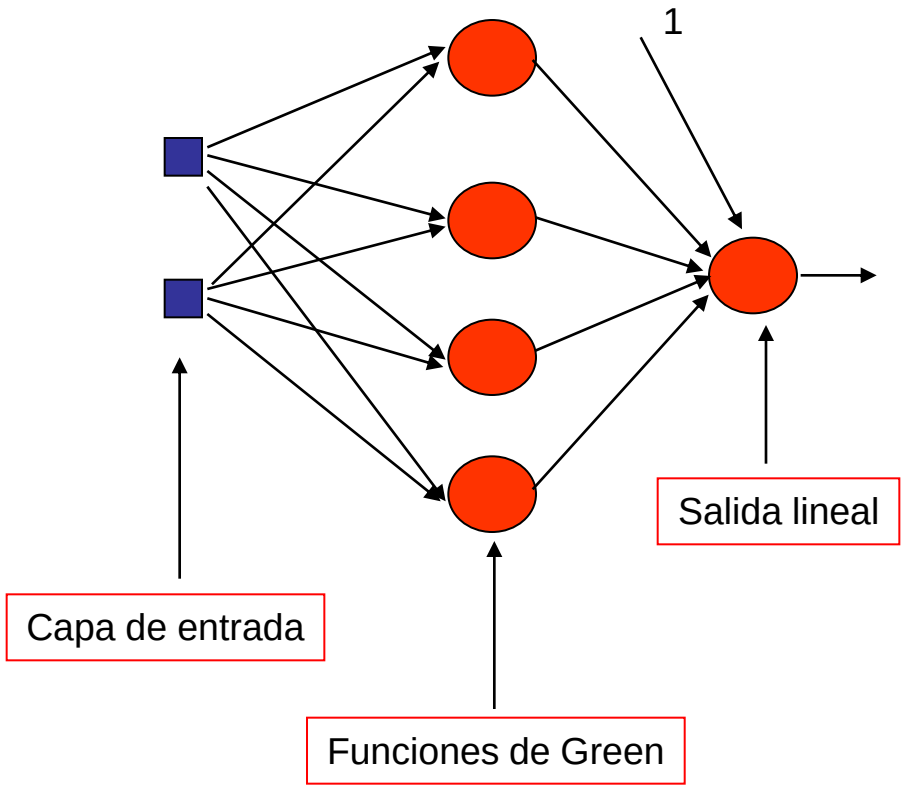
En esta ecuación las $G(x,x_i)$ son las funciones de Green asociadas con el operador diferencial L, definido como

$$L = D\tilde{D}$$

Las funciones de Green forman una base N-dimensional para la clase de funciones suaves. Si el operador diferencial D tiene las propiedad de ser translacionalmente invariante (depende solo de la distancia entre los argumentos) y rotacionalmente invariante (igual para cualquier radio o distancia), entonces

$$G(x, x_i) = G\left(\left\|x - x_i\right\|\right)$$

Redes Regularizadas



Si

$$D = \sum_n \sqrt{\frac{\sigma_i^{2n}}{n!2^n}} \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \quad \cdots \quad + \frac{\partial^2}{\partial x_{m_0}^2} \right)^n$$

Entonces,

$$G(x, x_i) = \exp\left(-\frac{1}{2\sigma_i^2} \|x - x_i\| \right)$$

$$w_i \text{ es el i-esimo elemento de } W = \left(G + \lambda I\right)^{-1} \vec{d}$$

con

$$G = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1N} \\ G_{21} & & & G_{2N} \\ \vdots & & & \vdots \\ G_{N1} & G_{N2} & \cdots & G_{NN} \end{bmatrix}$$

y $$G_{ij} = G(x_i, x_j) = G(x_j, x_i)$$

Además es positiva definida para ciertas clases de funciones (D), luego es invertible. Si λ =0, entonces G= Φ para el caso de funciones de green radiales.

GRBF

(Redes de base radial generalizadas)

Al igual que antes, una de las dificultades es el gran número de neuronas en la capa oculta, lo cual requiere de un gran número de operaciones para hacer la inversión de la matriz. (Dato: número de op. para invertir una matriz NxN es $O(N^3)$!!!!)

La idea es hacer una aproximación con una red sub-óptima, en el sentido de que el error cometido es distinto de cero.

Esto se logra reduciendo la complejidad de la red, haciendo la expansión sobre una familia de funciones (reducidas) linealmente independientes (como hicimos antes).

$$F^*(x) = \sum_{i=1}^{m_1} w_i \varphi_i(x) \quad \text{con } m_i < N$$

$$F^*(x) = \sum_{i=1}^{m_1} w_i \varphi_i(x) \quad \text{con } m_i < N$$

Una escogencia natural son la funciones de base radial

$$\varphi_i(x) = G\left(\|x - t_i\|\right)$$

Con los centros t_i a ser determinados. (Si $N=m_1$ y $t_i=x_i$, tenemos el aproximador universal).

Queremos minimizar;

$$E(F^*) = \sum_{i=1}^{N} \left[d_i - F^*(x_i)\right]^2 + \lambda \|DF\|^2$$

$$= \sum_{i=1}^{N} \left[d_i - \sum_{j=1}^{m_1} w_j G\left(\|x_i - t_j\|\right) \right]^2 + \lambda \|DF\|^2$$

Esto se logra con la resolución del siguiente sistema de ecuaciones:

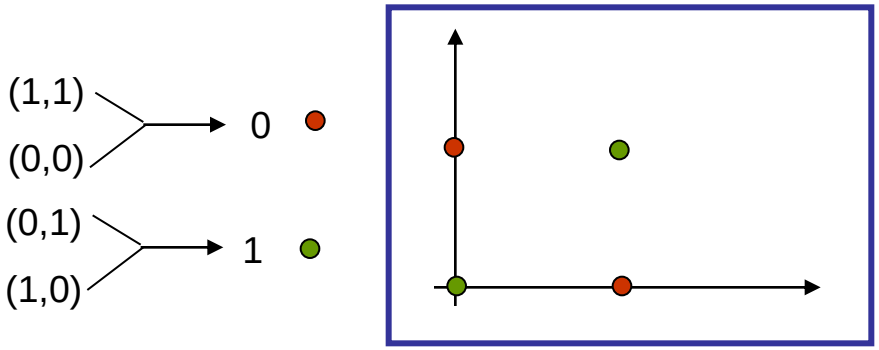
$$\left(G^T G + \lambda G_0\right) w = G^T d$$

Con:

$$G = \begin{bmatrix} G(x_1,t_1) & G(x_1,t_2) & \cdots & G(x_1,t_{m_1}) \\ G(x_2,t_1) & & & G(x_2,t_{m_1}) \\ \vdots & & & \vdots \\ G(x_N,t_1) & G(x_N,t_2) & \cdots & G(x_N,t_{m_1}) \end{bmatrix}$$

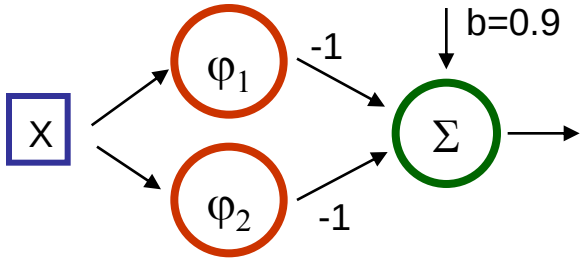
$$G_0 = \begin{bmatrix} G(t_1,t_1) & G(t_1,t_2) & \cdots & G(t_1,t_{m_1}) \\ G(t_2,t_1) & & & G(t_2,t_{m_1}) \\ \vdots & & & \vdots \\ G(t_{m_1},t_1) & G(t_{m_1},t_2) & \cdots & G(t_{m_1},t_{m_1}) \end{bmatrix}$$

Ejemplo:XOR

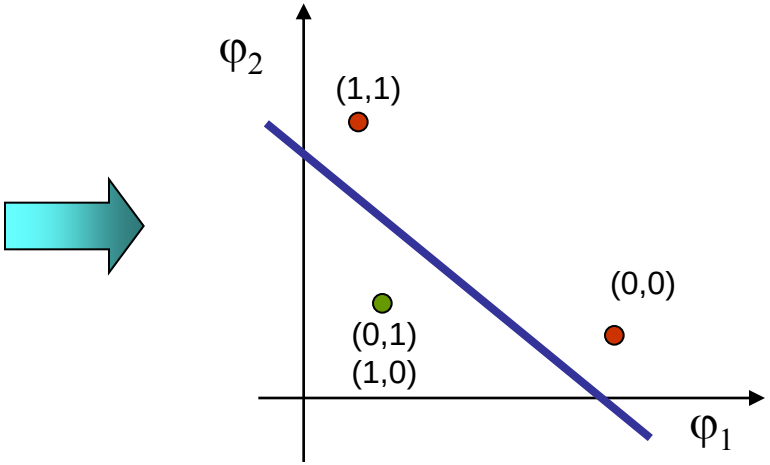


Sean $\varphi_1(x) = \exp(-\|x-x_1\|)$

$\varphi_2(x) = \exp(-\|x-x_2\|)$



	φ_1	φ_2
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678



Planteamos el sistema de ecuaciones….

$$G(\|x_1 - t_1\|)w_1 + G(\|x_1 - t_2\|)w_2 + b = d_1$$
$$G(\|x_2 - t_1\|)w_1 + G(\|x_2 - t_2\|)w_2 + b = d_2$$
$$G(\|x_3 - t_1\|)w_1 + G(\|x_3 - t_2\|)w_2 + b = d_3$$
$$G(\|x_4 - t_1\|)w_1 + G(\|x_4 - t_2\|)w_2 + b = d_4$$

RBF-OExcl.m

Estrategias para la escogencia de los centros

1. **Centros fijos seleccionados al azar.**

   - Mas sencillo de todos los métodos
   - **Tips:**
     - Escoger

$$G\left(\|x - t_i\|^2\right) = \exp\left(-\frac{m_1}{d_{\max}}\|x - t_i\|^2\right)$$

Con m_1 =número de centros,

d_{max}=distancia máxima entre los centros

- Usar centros con gausianas mas anchas donde hay poca data (requiere hacer experimentación con la data)

- Experimentar con distintos números de neuronas.

2. Selección de centros por clusters

• Busca centros apropiados donde existe similaridad en data y agruparlos. Se quiere lograr que los centros sean aproximadamente la media del “cluster” . Hacer la actualización de los pesos.

• **Algortimo**

1. **Escoger m_1 centros aleatoriamente, $\{t_k(0)\}$, $k=1,\ldots,m_1$**
2. **Mientras $||t_k(n+1)-t_k(n)|| <$ tolerancia**
    1. **Escoger patron x al azar**
    2. **$K(x)=\text{argmin}\{||x - t_k(n)||\}$, $k=1,\ldots,m_1$**
    3. **$t_k(n+1)=\begin{cases} t_k(n) + \alpha\,[\,x - t_k(n)\,], & k=k(x) \\ t_k(n) & \text{caso contrario} \end{cases}$**

Luego puedo usar la pseudoinversa para calcular los pesos o usar LMS pensando en estas como entradas a un ADALINE.

3. Selección supervisada de centros.

Idea: Adaptar los parametros libres (pesos,centros, y parametros de dispersion) usando LMS (descenso de gradiente) para minimizar

$$\mathrm{E} = \frac{1}{2}\sum_{j=1}^{N} e_j^2 = \frac{1}{2}\sum_{j=1}^{N}\left(d - \sum_{i=1}^{M} w_i G\left(\left\| x_j - t_i \right\|\right)\right)^2$$

Cada actualización requiere del gradiente de la función con respecto al parámetro en cuestión.

Actualización de los pesos

$$\frac{\partial E(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n) G\left(\left\| x_j - t_i(n) \right\|\right)$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial E(n)}{\partial w_i(n)}$$

Actualización de los centros

$$\frac{\partial E(n)}{\partial t_i(n)} = w_i(n) \sum_{j=1}^{N} e_j(n) G'\left(\left\| x_j - t_i(n) \right\|\right) \frac{\left[x_j - t_i(n)\right]}{2\left\| x_j - t_i \right\|}$$

$$t_i(n+1) = t_i(n) - \eta_2 \frac{\partial E(n)}{\partial t_i(n)}$$

Para las dispersiones

$$\frac{\partial E(n)}{\partial \sigma_i(n)} = -w_i(n)\sum_{j=1}^{N} e_j(n) G'\left(\left\| x_j - t_i(n) \right\|\right)\sigma_i^{-1}(n)$$

$$\sigma_i(n+1) = \sigma_i(n) - \eta_3 \frac{\partial E(n)}{\partial \sigma_i(n)}$$

Se hace tan computacionalmente intensivo como las MLP!!

Se puede usar un entrenamiento como los descrito anteriormente y hacer la entonación de los parámetros con este proceso.

4. **Selección no-supervisada de centros (redes Kohonen).**

Se representan los datos en una malla y se ejecuta un algoritmo auto-organizativo para determinar los centros.

RBF	MLP
Tiene una sola capa oculta	Puede tener mas de una capa oculta
La función de transferencia en la capa oculta es distinta (dependencia de centros)	Tipicamente las neuronas en las capas escondidas tienen la misma neurona
Capa oculta es no lineal mientras que la de salida es lineal	Las MLP todas las neuronas pueden ser nolineales (puede variar según aplicación)
El argumento de capa oculta calcula norma euclidea entre entrada y centro	La función de activación recibe el producto interno entre los pesos y la entrada

Como calcular λ

Sea $V(\lambda) = \dfrac{\frac{1}{N}\left\|I - A(\lambda)y\right\|^2}{\left[\frac{1}{N}tr\left[I - A(\lambda)\right]\right]^2}$ **con** $\begin{cases} F_\lambda = A(\lambda)y \\ \left(F_\lambda(x_k) = \sum_{i=1}^{N} a_{ki}(\lambda) y_i \right) \end{cases}$

El λ que minimiza V esta suficientemente cercano al λ que minimiza

$$R(\lambda) = \frac{1}{N}\sum_{i=1}^{N}\left[f(x_i) - F_\lambda(x_i)\right]^2$$

cuando queremos aproximar

$$y_i = f(x_i) + \in_i$$