Tema 3: Aprendizaje Automático en Máquinas

Universidad Pontificia de Salamanca

Manuel Martín-Merino

Contenido

- Aprendizaje estadístico: Introducción
- Aprendizaje estadístico: Sesgo-Varianza
- Problemas dentro del aprendizaje automático
- Análisis discriminante: Modelos lineales
- Análisis discriminante probabilístico
- Redes Neuronales: Modelos no lineales
- Estimación de errores y comparación modelos
- Tópicos avanzados
- Resúmen: Discusión

Aprendizaje estadístico: Introducción (I)

- Sea $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ una muestra finita de objetos donde $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ es la representación vectorial del objeto i y y_i es la salida óptima.
- El problema de **aprendizaje automático** trata de encontrar una función $f : \mathcal{X} \to \mathbb{R}$ que permita *predecir la salida para nuevas observaciones* con error mínimo.
- $y_i \in \mathbb{R}$ en problemas de predicción mientras que $y_i \in \{\pm 1\}$ en problemas de clasificación.

Aprendizaje estadístico: Introducción (II)

Matemáticamente, dada una muestra de datos $(\boldsymbol{x}_i, y_i)$ independientes e idénticamente distribuidos según $p(\boldsymbol{x}, y)$ desconocida, tratamos de encontrar f.

Esto requiere:

- Proponer una familia de funciones aproximadoras suficientemente flexible $f(\boldsymbol{x}; w)$.
- Proponer una función de error $L(y, f(\boldsymbol{x}; \omega))$ que mida la calidad de la aproximación.
- Minimizar el error promedio $E(L(y, f(\boldsymbol{x}; \omega)))$.

Aprendizaje estadístico: Introducción (III)

Objetivo: Aprender la función óptima $f(\boldsymbol{x};\omega)$ que minimiza:

$$E(w) = \int L(y, f(\boldsymbol{x};\omega))p(\boldsymbol{x}, y)\, d\boldsymbol{x}\, dy \tag{1}$$

como $p(\boldsymbol{x}, y)$ es desconocida, el riesgo funcional se aproxima por el empírico más un término proporcional a la complejidad del modelo:

$$E(\omega) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i;\omega)) + \lambda\Omega(\omega) \tag{2}$$

Aprendizaje estadístico: Introducción (IV)

Objetivo: Encontrar una función $f(\boldsymbol{x}_i; \omega)$ que obtenga un balance entre error de entrenamiento y complejidad del modelo.

- $f(\boldsymbol{x}_i; \omega)$ muy compleja implica error entrenamiento 0 pero $\Omega(\omega)$ grande.
- $f(\boldsymbol{x}_i; \omega)$ muy sencilla implica minimizar $\Omega(\omega)$ a costa de error de entrenamiento grande.

Aprendizaje estadístico: Sesgo-Varianza (I)

La complejidad del modelo debe buscar un balance entre sesgo-varianza.

Consideramos que existen varios conjuntos de entrenamiento $\mathcal{D}$ con n patrones y obtenidos de la misma distribución de probabilidad $p(\boldsymbol{x}, y)$.

Por ser la muestra finita $f_{\mathcal{D}}(\boldsymbol{x}; \omega)$ dependerá de la muestra particular elegida. Por ello el error se calcula como un promedio para todas las muestras de entrenamiento

$$\mathcal{E}_{\mathcal{D}}\{f_{\mathcal{D}}(\boldsymbol{x}; \omega) - g(\boldsymbol{x}))^2\} \tag{3}$$

Aprendizaje estadístico: Sesgo-Varianza (II)

Este error se puede descomponer como:

$$\mathcal{E}_{\mathcal{D}}[\{f_{\mathcal{D}}(\boldsymbol{x};\omega) - g(\boldsymbol{x})\}^2] =$$
$$\underbrace{\{\mathcal{E}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{x};\omega)] - g(\boldsymbol{x})\}^2}_{\text{sesgo}^2} + \underbrace{\mathcal{E}_{\mathcal{D}}[\{f_{\mathcal{D}}(\boldsymbol{x};\omega) - \mathcal{E}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{x};\omega)]\}^2]}_{\text{varianza}} \quad (4)$$

- *Sesgo*: Mide la desviación de la función sobre el valor óptimo del promedio de las estimaciones para diferentes conjuntos de entrenamiento.
- *Varianza*: Mide la sensibilidad del estimador de la muestra a la muestra particular elegida.

Aprendizaje estadístico: Sesgo-Varianza (III)

Ejemplo: *Consideramos el problema de regresión* $y = (\text{sen}\, 2\pi x)^2 + \epsilon$, *donde* ϵ *es una variable aleatoria de media* 0 *y* $\sigma_\epsilon = 0{,}2$. *El tamaño del conjunto de entrenamiento es* $n = 30$

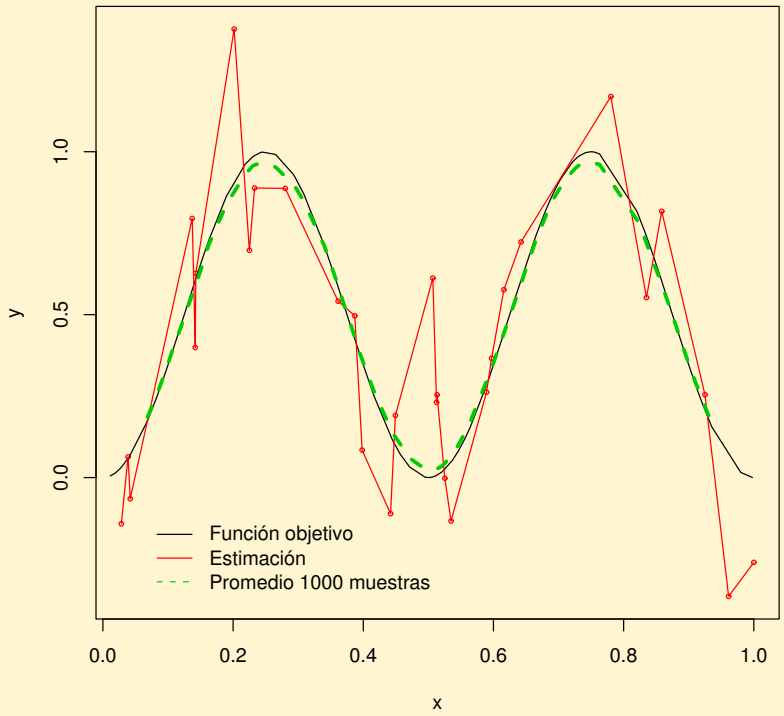


Fig. 1: Grados de libertad 30. Varianza elevada.

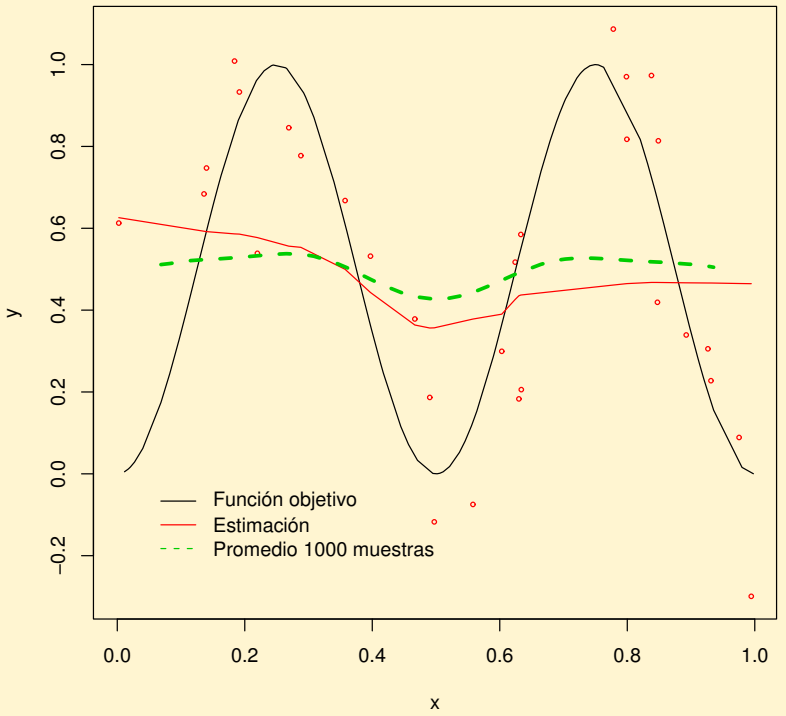
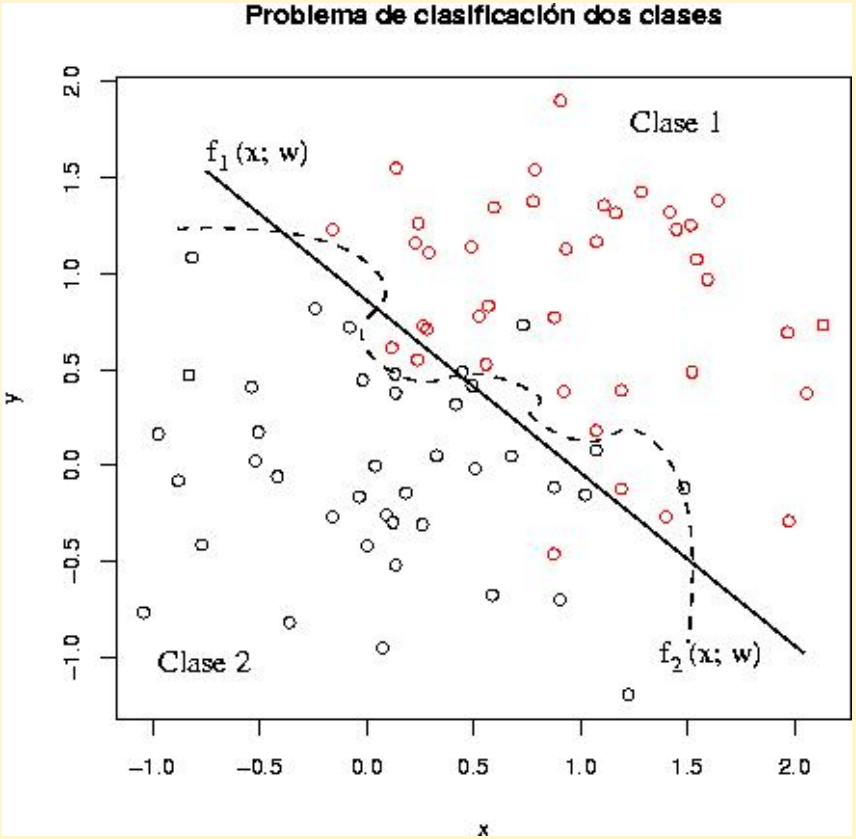


Fig. 2: Grados de libertad $2{,}39$. Sesgo elevado.

Problemas dentro del aprendizaje automático (I)

Clasificación



Ejemplo clasificación

- Proponer una familia de funciones (por ej. polinomios)
- Proponer una función de error (por ej. cuadrático)
- Estimar los parámetros: Optimizar

Clasificación

Objetivo: Estimar la $p(C_j|\boldsymbol{x})$ para nuevos patrones no utilizados en la fase de entrenamiento.

Enfoques:

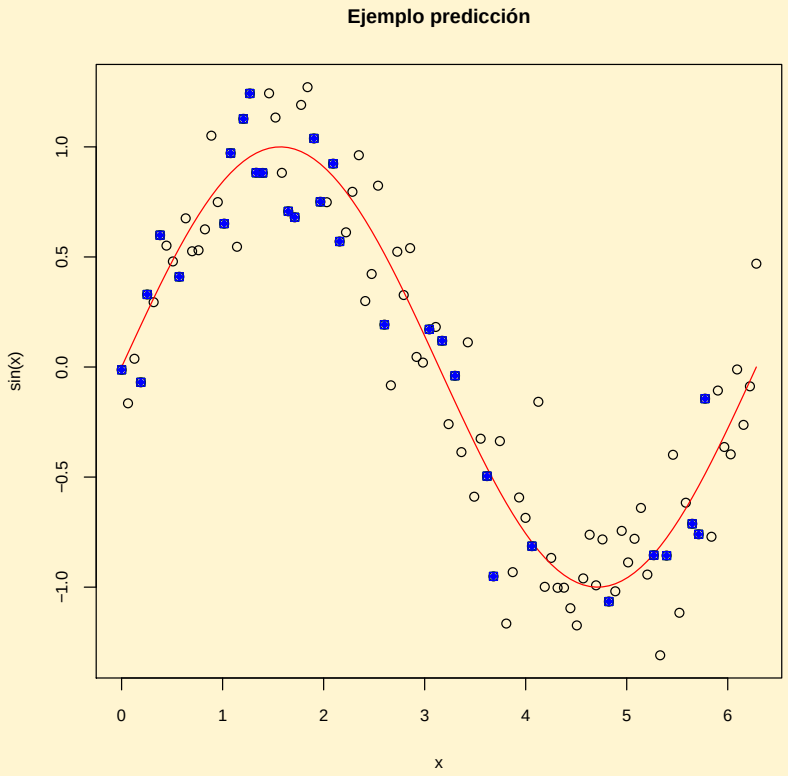
- Probabilístico. Estima $p(\boldsymbol{x}|C_j)$ y aplica la regla de decisión Bayesiana:

$$\boldsymbol{x} \in C_k \quad \text{si} \quad p(\boldsymbol{x}|C_k)p(C_k) > p(\boldsymbol{x}|C_j)p(C_j)\,\forall j \neq k \qquad (5)$$

- Redes Neuronales y regresión logística. Estima directamente $p(C_j|\boldsymbol{x})$
- Análisis discriminante. Estima directamente la ecuación de la frontera $f(\boldsymbol{x};\omega)$

Problemas dentro del aprendizaje automático (II)

Predicción: Identificación funciones

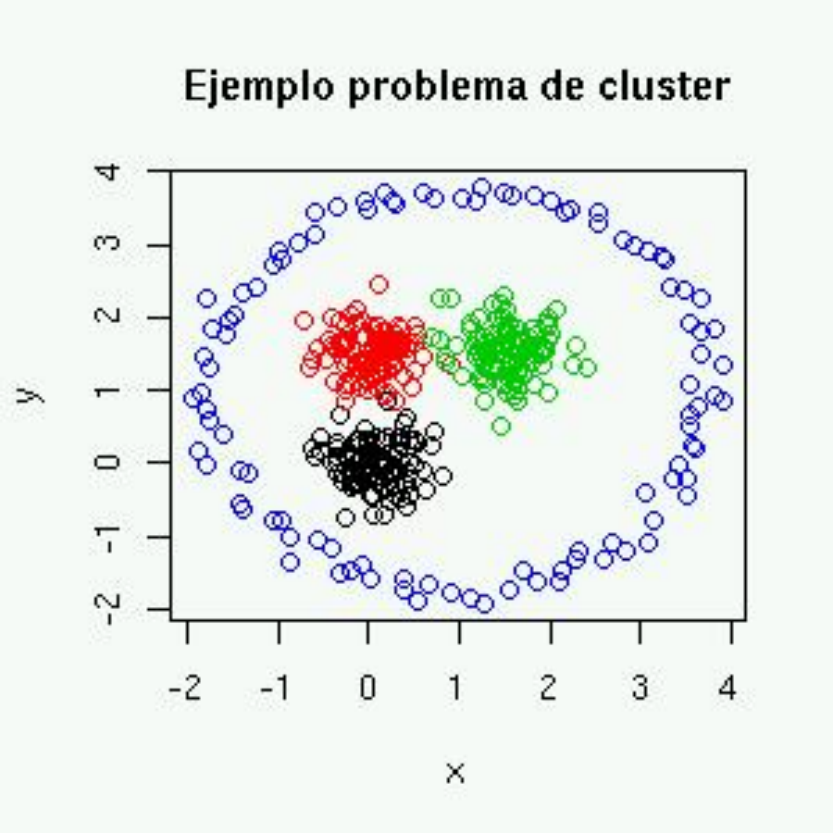


Ejemplo identificación funciones

- Proponer una familia de funciones (por ej. polinomios)
  - Modelos lineales: Regresión lineal, logística.
  - Modelos no lineales: Redes Neuronales.
- Proponer una función de error (por ej. cuadrático)
- Estimar los parámetros: Optimizar

Problemas dentro del aprendizaje automático (III)

Análisis de Cluster

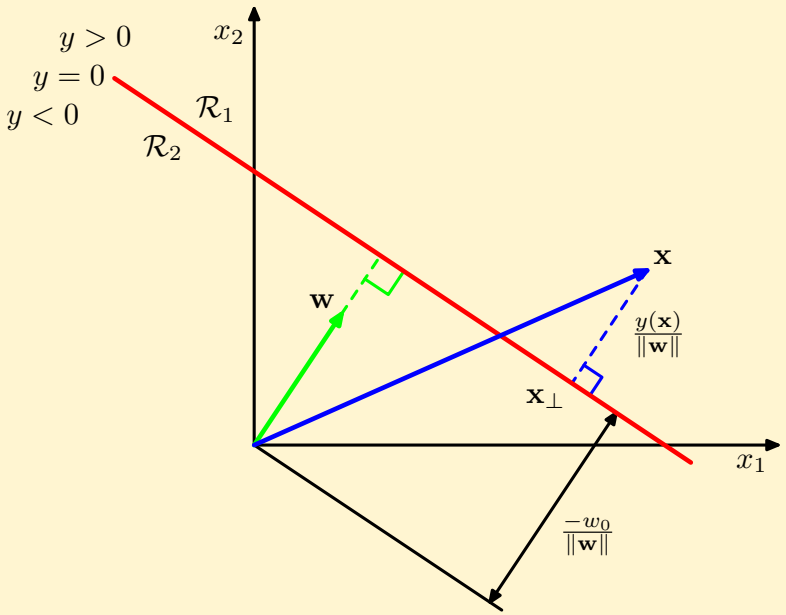


Ejemplo *cluster*

- Proponer un modelo
- Definir función de error que mida calidad cluster
- Estimar los parámetros: Optimizar

Análisis discriminante: Modelos lineales (I)

- Sea $\{\boldsymbol{x}_i, \boldsymbol{t}_i\}_{i=1}^N$ una muestra finita de patrones de entrenamiento y $\{C_j\}_{j=1}^M$ el conjunto de clases. $t_{ij} = 1$ si $\boldsymbol{x}_i$ pertenece a la clase C_j.

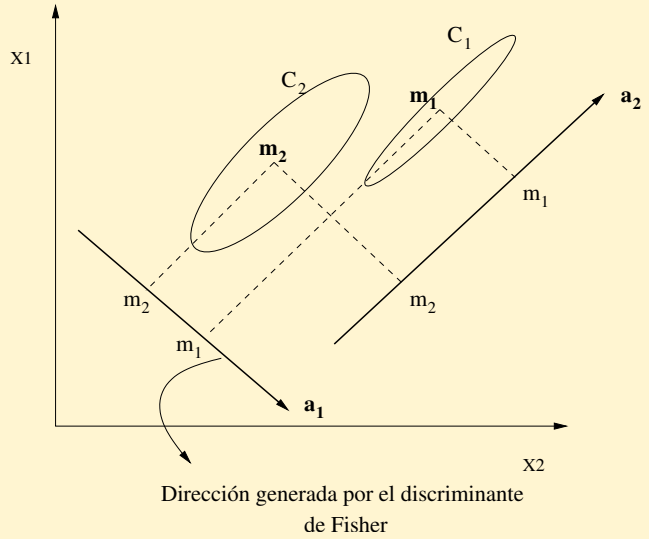


Hiperplano discriminante lineal

- La función discriminante lineal se escribe como: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$
  $f(\boldsymbol{x}) \geq 0$ entonces $\boldsymbol{x} \in C_1$, en otro caso $\boldsymbol{x} \in C_2$.
- ¿ Función de error a optimizar ?

Análisis discriminante: Modelos lineales (II)

Enfoque Fisher: Proyecta los datos sobre un hiperplano lineal que maximiza la separabilidad entre clases.



Criterio de separabilidad (biclase):

- Maximiza separación entre centros de clases proyectadas $(\bar{m}_2 - \bar{m}_1)^2$.
- Minimiza la dispersión de cada clase en torno a su media $(s_1^2 + s_2^2)$ donde $s_k^2 = \sum_{i \in C_k} (\bar{x}_i - \bar{m}_k)^2$.

Criterio de Fisher

La función de error a optimizar se puede escribir como:

$$J(\boldsymbol{w}) = \frac{(\bar{m}_2 - \bar{m}_1)^2}{s_1^2 + s_2^2} = \tag{6}$$

$$= \frac{\boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T\boldsymbol{w}}{\sum_{x_i \in C_1} \boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^T\boldsymbol{w} + \sum_{x_i \in C_2} \boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{m}_2)(\boldsymbol{x}_i - \boldsymbol{m}_2)^T\boldsymbol{w}} \tag{7}$$

Definimos las matrices de covarianza interclase e intraclase como:

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \tag{8}$$

$$S_w = \sum_{x_i \in C_1} \boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^T\boldsymbol{w} + \sum_{x_i \in C_2} \boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{m}_2)(\boldsymbol{x}_i - \boldsymbol{m}_2)^T\boldsymbol{w} \tag{9}$$

Análisis discriminante: Modelos lineales (III)

Esto supone optimizar el siguiente criterio:

$$J_F = \frac{\mid \mathbf{w}^T S_b \mathbf{w} \mid}{\mid \mathbf{w}^T S_w \mathbf{w} \mid}, \tag{10}$$

Esto equivale a resolver el siguiente problema de autovalores y autovectores:

$$S_b \mathbf{w} = S_w \mathbf{w} \Lambda \tag{11}$$

que se puede resolver eficientemente mediante operaciones de álgebra lineal.

Proyectados los datos, se asignan a la clase de centroide más cercano.

Análisis discriminante: Modelos lineales (IV)

Propiedades discriminante Fisher:

- Es atractivo para trabajar en alta dimensión y eficiente computacionalmente.
- La frontera inducida es lineal y no está preparado para trabajar con clases multimodales.
- La dimensión máxima del espacio proyección es $M - 1$.
- Es bastante sensible a atípicos y no maximiza la capacidad de generalización.

Análisis discriminante: Modelos lineales (V)

Discriminante lineal puede **optimizar** también el **error cuadrático medio**:

$$E = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i + w_0 - t_i)^2 \tag{12}$$

Definimos la variable respuesta:

$$t_i = \begin{cases} N/N_1 & \text{si} \quad \boldsymbol{x_i} \in C_1 \\ -N/N_2 & \text{si} \quad \boldsymbol{x_i} \in C_2 \end{cases} \tag{13}$$

Calculando las derivadas e igualando a 0 obtenemos:

$$\frac{\partial E}{\partial w_0} = \sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i + w_0 - t_i) = 0 \tag{14}$$

$$\frac{\partial E}{\partial \boldsymbol{w}} = \sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i + w_0 - t_i)\boldsymbol{x}_i = 0 \tag{15}$$

(16)

Operando se obtiene:

$$(\boldsymbol{S}_w + \frac{N_1 N_2}{N}\boldsymbol{S}_B)\boldsymbol{w} = N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{17}$$

Teniendo en cuenta que $\boldsymbol{S}_B\boldsymbol{w}$ es paralelo a $\boldsymbol{m}_2 - \boldsymbol{m}_1$ el vector $\boldsymbol{w}$ de la función discriminante se puede expresar:

$$\boldsymbol{w} \propto \boldsymbol{S}_w^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{18}$$

que es equivalente a la expresión obtenida maximizando la separabilidad.

LDA: Optimización error cuadrático (I)

- Muchos algoritmos de aprendizaje minimizan el error cuadrático medio definido como:

$$E_D(\omega) = \sum_{i=1}^{N} (f(\boldsymbol{x}_i; \omega) - t_i)^2 \,, \tag{19}$$

donde $f(\boldsymbol{x}_i; \omega) = \boldsymbol{w}^T \boldsymbol{x}_i$ en el análisis discriminante lineal.

- Sea $\mathbf{X} = (x_{ij})$ la matriz de dimensión $n \times (d+1)$ con fila i-ésima el patrón $\boldsymbol{x}_i \in \mathbb{R}^{(d+1)}$. El error cuadrático es expresa matricialmente:

$$E_D(\omega) = \|\mathbf{X}\boldsymbol{w} - \boldsymbol{t}\|^2 \tag{20}$$

- Calculando la derivada de la función de error e igualando a 0 obtenemos:

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{i=1}^{n} 2(\boldsymbol{w}^T \boldsymbol{x}_i - t_i)\boldsymbol{x}_i = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{w} - \boldsymbol{t}) = 0 \qquad (21)$$

  Esto da lugar a la ecuación:

$$\mathbf{X}^t\mathbf{X}\boldsymbol{w} = \mathbf{X}^T\boldsymbol{t} \qquad (22)$$

La expresión anterior permite obtener una solución para $\mathbf{w}$:

$$\boldsymbol{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{t} = \mathbf{X}^\dagger\boldsymbol{t} \qquad (23)$$

donde $\mathbf{X}^\dagger$ es la pseudoinversa de la matriz $\mathbf{X}$.

LDA: Error cuadrático regularizado (II)

- La función de error (19) no considera la capacidad de generalización del hiperplano. La solución puede sobreajustar los datos incrementando el error de predicción sobre el test.
- Para evitar este problema se introduce un término de regularización proporcional a la complejidad del modelo.

$$E(\boldsymbol{w}) = \underbrace{\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - t_i)^2)}_{E_D(\boldsymbol{w}} + \frac{\lambda}{2}\underbrace{\boldsymbol{w}^T\boldsymbol{w}}_{E_w(\boldsymbol{w}} \tag{24}$$

- La optimización de dicha función de error favorece que algunos pesos converjan a 0 reduciendo el número parámetros.

- La solución para el vector director del hiperplano se expresa ahora como:

$$\boldsymbol{w} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{t} \tag{25}$$

- Otros términos de regularización de la forma $E_w(\boldsymbol{w} = \sum_{j=1}^{d} |w_j|^q$ generan modelos con diferentes propiedades.

LDA: Aprendizaje incremental (III)

- En algunas aplicaciones los datos van llegando uno a uno. Los parámetros del modelo se deben adaptar después de recibir cada dato de forma incremental.

- La función de error se optimiza mediante un algoritmo iterativo **gradiente descendente estocástico**.

  - Si el error descompone como suma de los errores para cada patrón n, $E = \sum_n E_n$, el vector de pesos se adapta utilizando la siguiente regla:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \nabla E_n \tag{26}$$

  - Para error cuadrático medio se obtiene la siguiente regla:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta(t_n - \boldsymbol{w}^T(t)\boldsymbol{x}_n)\boldsymbol{x}_n \tag{27}$$

    Este algoritmo se conoce como LMS.

Análisis discriminante probabilistico (I)

Idea: Trata de resolver mediante una regresión lineal problemas de clasificación fuertemente no lineales.

- Asumimos que el cociente de probabilidades sigue una curva lineal:

$$\log\left(\frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)}\right) = w_0 + \boldsymbol{w}^T\boldsymbol{x} \tag{28}$$

- Aplicando la regla de Bayes en (28) es fácil probar:

$$p(C_1|\boldsymbol{x}) = \frac{1}{1+e^{-a}} = \sigma(a) \quad (29)$$

$$p(C_2|\boldsymbol{x}) = \frac{1}{1+e^{a}} = \sigma(-a) \quad (30)$$

donde $a = \boldsymbol{w}^T\boldsymbol{x} + w_0'$ y $w_0' = w_0 + \ln\frac{p(C_1)}{p(C_2)}$. σ es la función sigmoide y juega un papel importante en problemas de clasificación.

Análisis discriminante probabilistico (II)

Estimación de los parámetros

- Maximizando la verosimilitud logarítmica (Estadística)

  Sea $\{\boldsymbol{x}_i, t_i\}$ una muestra finita de observaciones con $t_i \in \{0, 1\}$ la etiqueta para el patrón $\boldsymbol{x}_i$. La verosimilitud se escribe:

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{i=1}^{N} y_i^{t_i}(1 - y_i)^{1-t_i}\,, \tag{31}$$

  con $y_i = p(C_1|\boldsymbol{x}_i)$.

Tomando logaritmos obtenemos la función de error a optimizar:

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}|\boldsymbol{w}) = -\sum_{i=1}^{N}\{t_i \ln y_i + (1-t_i)\ln(1-y_i)\} \quad (32)$$

Esta función de error está más justificada para clasificación que el error cuadrático medio y se llama entropía cruzada.

Análisis discriminante probabilistico (III)

Estimación de los parámetros

- Minimizando el error cuadrático medio (Redes neuronales)

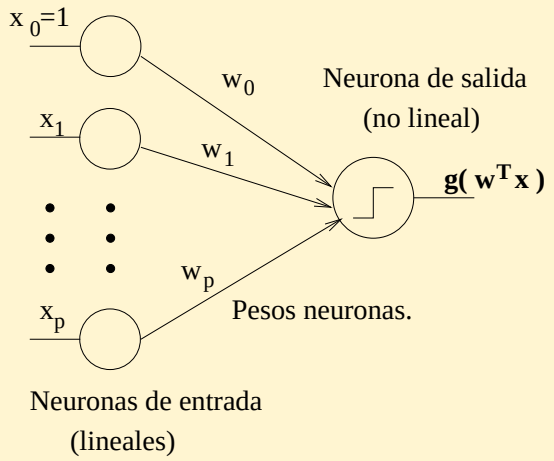
Redes neuronales: Modelos no lineales semi-paramétricos

Contenido

- Perceptrón monocapa
- Perceptrón multicapa: Algoritmo Backpropagation
- Redes RBF
- Tópicos avanzados

Perceptrón monocapa (I)

Arquitectura



Sea $\boldsymbol{x} = (-1, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$ y $\boldsymbol{w} = (w_0, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$ el vector de pesos de la neurona. La salida de la neurona vale:

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{d} w_i x_i \geq w_0 \Longrightarrow \quad \boldsymbol{w}^T \boldsymbol{x} \geq 0 \\ -1 & \text{si} \quad \sum_{i=1}^{n} w_i x_i < w_0 \Longrightarrow \quad \boldsymbol{w}^T \boldsymbol{x} < 0 \end{cases} \tag{33}$$

Se pueden utilizar otras funciones de activación no lineales que sean continuas como:

$$g(\boldsymbol{x}) = \frac{1}{1+e^{-k\boldsymbol{x}}} \quad \text{función de activación sigmoide} \quad (34)$$

$$g(\boldsymbol{x}) = \frac{e^{k\boldsymbol{x}}-e^{-k\boldsymbol{x}}}{e^{k\boldsymbol{x}}+e^{-k\boldsymbol{x}}} \quad \text{función tangente hiperbólica} \quad (35)$$

Perceptrón monocapa (II)

- Función de error:
  - Porcentaje de datos mal clasificados: No es continua y constante a trozos.
  - Promedio de las "distancias" de los patrones mal clasificados al hiperplano separador:

$$J(\boldsymbol{w}) = -\sum_{x_i \in \mathcal{M}} \boldsymbol{w}^T \boldsymbol{x}_i t_i \tag{36}$$

donde $\mathcal{M}$ es el conjunto de patrones mal clasificados por la red neuronal y $\boldsymbol{w}^T \boldsymbol{x}_i$ determina la distancia del patrón $\boldsymbol{x}_i$ al hiperplano separador.

- Algoritmo de optimización: Gradiente descendente (estocástico para versión incremental).

Reglas de adaptación pesos

La regla del gradiente descendente se expresa:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta(t)\nabla J_p(\boldsymbol{w}) \tag{37}$$

Algoritmo de aprendizaje Batch

1: Inicializar $\boldsymbol{w}$, η, θ y $t \leftarrow 0$
2: **repeat**
3: $\quad t \leftarrow t+1$
4: $\quad \boldsymbol{w}(t) \leftarrow \boldsymbol{w}(t-1) + \eta(k) \sum_{x_i \mathcal{M}} \boldsymbol{x_i}$
5: **until** $|\eta(t) \sum_{x_i \mathcal{M}} \boldsymbol{x_i}| < \theta|$
6: **return** $\boldsymbol{w}$
7: End

Algoritmo de aprendizaje incremental

1: Inicializar $\boldsymbol{w}$, η, θ y $t \leftarrow 0$
2: **repeat**
3: $\quad t \leftarrow t + 1$
4: $\quad \boldsymbol{w}(t) \leftarrow \boldsymbol{w}(t-1) + \eta(t)\boldsymbol{x}(t)$
5: **until** $|\eta(t)\boldsymbol{x}(t)| < \theta|$
6: **return** $\boldsymbol{w}$
7: End

Teorema de convergencia: Se puede demostrar que si el conjunto de entrenamiento es linealmente separable entonces el algoritmo de aprendizaje del perceptrón converge a una solución en un número finito de pasos.

Otras funciones de error:

- Se pueden optimizar múltiples funciones de error que dan soluciones con diferentes propiedades.
- Si optimizamos el error cuadrático medio $J_c(\boldsymbol{w}) = \|\mathbf{X}\boldsymbol{w} - \boldsymbol{t}\|^2$ se obtiene aplicando el grandiente descentente estocástico (27) el siguiente algoritmo iterativo:

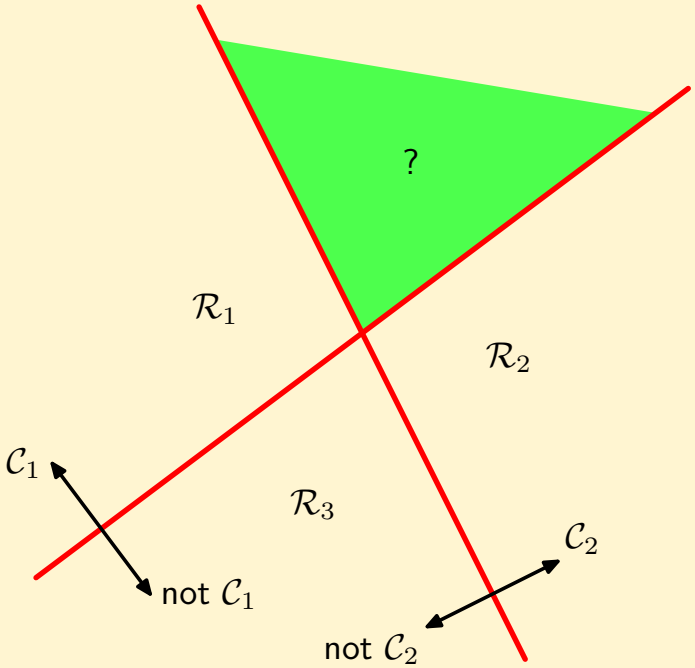
1: Inicializar $\boldsymbol{w}$, η, θ y $k \leftarrow 0$
2: **repeat**
3: $\quad k \leftarrow k + 1$
4: $\quad \boldsymbol{w}(k) \leftarrow \boldsymbol{w}(k-1) + \eta(k)[t(k) - \boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}(k)]$
5: **until** $|\eta(k)\boldsymbol{x}(k)| < \theta|$
6: **return** $\boldsymbol{w}$
7: End

Perceptrón monocapa (III)

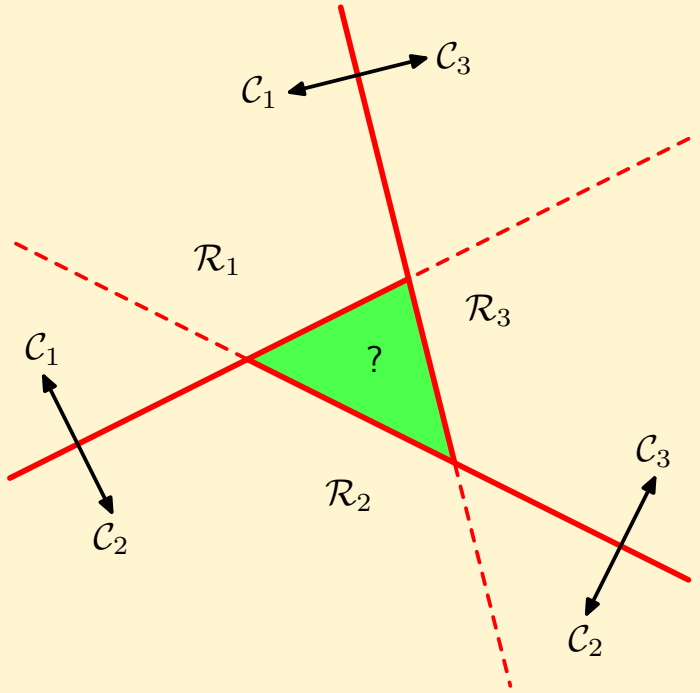
Extensión al caso multiclase

- Estrategia "uno frente al resto". Construir $K-1$ clasificadores que resuelven problemas biclase.
- Estrategia "uno frente a uno". Construir $K(K-1)/2$ clasificadores binarios y aplicar estrategia voto.
- Construir K discriminantes $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$ y asignar $\boldsymbol{x} \in C_k$ si $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x}) \, \forall j \neq k$.

Problemas estrategias multiclase



Estrategia uno frente al resto



Estrategia uno frente a uno

Perceptrón monocapa (IV)

limitaciones perceptrón monocapa

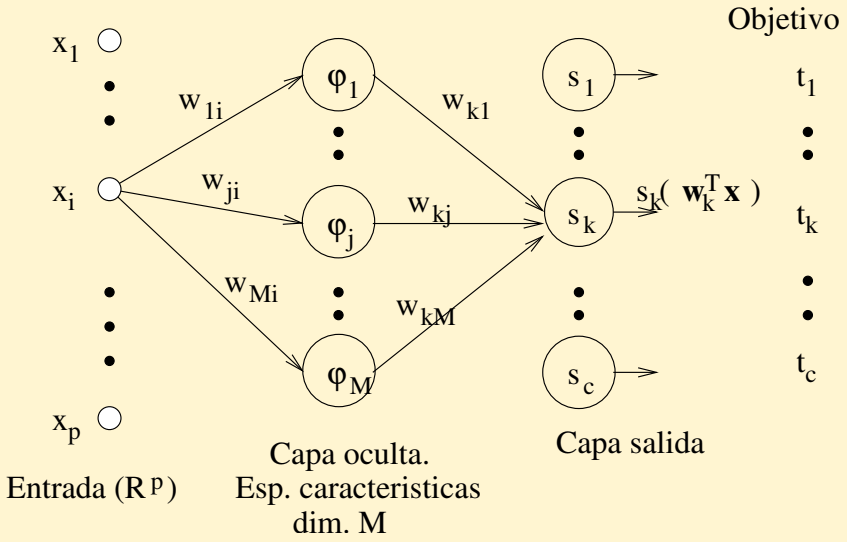
- Si las clases no son convexas, el algoritmo no encuentra la solución.
- Si las clases son no lineales, el error será elevado.
  Solución:
  - ▷ Transformar no linealmente los datos para conseguir que sean separables en $\mathbb{R}^m$

$$\phi: \quad \mathbb{R}^d \rightarrow \mathbb{R}^m \tag{38}$$

$$\phi(\boldsymbol{x}) \rightarrow (\phi_1(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x})) \tag{39}$$

Perceptrón multicapa (I)

Arquitectura



- Las neuronas de la capa oculta realizan un análisis discriminante no lineal para convertir los datos en linealmente separables.

La salida de las neuronas de la capa oculta vale:

$$o_j = f_j(\sum_{i=1}^{d} w_{ij}x_i) = f_j(\boldsymbol{w}_j\boldsymbol{x}) \tag{40}$$

donde f_j es una función de activación no lineal. Normalmente en el perceptrón f_j es la función sigmoide.

Perceptrón multicapa (II)

- Las neuronas de salida implementan un análisis discriminante lineal probabilístico que separan las clases en el espacio generado por las neuronas de la capa oculta. La salida vale:

$$o_k = f_k(\sum_{j=1}^{M} w_{jk} o_j) = f_k(\boldsymbol{w}_k^T \boldsymbol{o}^{(j)}) \tag{41}$$

- Si la función de activación de salida f_k es sigmoide, se puede interpretar como una probabilidad $f_k(\boldsymbol{x}; \boldsymbol{w}) = p(C_k|\boldsymbol{x})$.

- Si la función de activación de salida es lineal se aplica a problemas de predicción.

Perceptrón multicapa (III)

- Función a optimizar: Error cuadrático medio. El error para el patrón $\boldsymbol{x}(p)$ vale:

$$E(p) = \frac{1}{2}\sum_{k=1}^{c}(t_k(p) - o_k(p))^2 \tag{42}$$

  Cuando se dispone de todos los patrones al mismo tiempo se hace un promedio de errores:

$$E = \frac{1}{N}\sum_{p=1}^{N} E(p) = \frac{1}{N}\sum_{p=1}^{N}\sum_{k=1}^{c}(t_k(p) - o_k(p))^2 \tag{43}$$

- Algoritmo optimización: Gradiente descendente (estocástico para versión incremental):

Perceptrón multicapa (IV)

- Notación:

$$e_k = \sum_{j=1}^{M} w_{jk} o_j \qquad o_k = f_k(e_k) \qquad f(e_k) = \frac{1}{1+\exp(-e_k)} \tag{44}$$

$$e_j = \sum_{i=1}^{d} w_{ij} x_i \qquad o_j = f_j(e_j) \qquad f(e_j) = \frac{1}{1+\exp(-e_j)} \tag{45}$$

- Adaptación de los pesos de la capa de salida w_{kj}

$$w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E}{\partial w_{jk}} \tag{46}$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial e_k}\frac{\partial e_k}{\partial w_{jk}} = -\delta_k o_j$$

$$\frac{\partial E}{\partial e_k} = -\delta_k = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial e_k} = (t_k - o_k) f_k'(e_k)$$

$$\boxed{w_{jk}(t+1) = w_{jk}(t) - \eta(t_k - o_k) f_k'(e_k) o_j}$$

- Adaptación de los pesos de la capa oculta w_{ji}

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial e_j}\frac{\partial e_j}{\partial w_{ij}} = \left(\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial e_j}\right)\boldsymbol{x}_i = \underbrace{\frac{\partial E}{\partial o_j} f_j'(e_j)}_{\delta_j}\boldsymbol{x}_i$$

$$\frac{\partial E}{\partial o_j} = \sum_{k=1}^{M}\frac{\partial E}{\partial e_k}\frac{\partial e_k}{\partial o_j} = \sum_{k=1}^{M}\frac{\partial E}{\partial e_k} w_{jk} = \sum_{k=1}^{M}\delta_k w_{jk}$$

$$w_{ij}(t+1) = w_{ij}(t) - \eta \boldsymbol{x}_i \underbrace{f_j'(e_j) \sum_{k=1}^{M} \delta_k w_{jk}}_{\delta_j}$$

Algoritmo “backpropagation” estocástico:

1: Inicializar $\boldsymbol{w}$, η, θ, M y $t \leftarrow 0$
2: **repeat**
3: $\quad t \leftarrow t + 1$
4: $\quad$ Sea $\boldsymbol{x}_t$ un patrón escogido aleatoriamente
5: $\quad w_{kj}(t) = w_{kj}(t-1) + \eta \delta_k o_j$
$\quad w_{ij}(t) = w_{ij}(t-1) + \eta \delta_j \boldsymbol{x}_i$
6: **until** $\|\nabla J(\boldsymbol{w})\| < \theta$
7: **return** $\boldsymbol{w}$
8: End

Perceptrón multicapa (V)

Aspectos prácticos

- Determinación del número de neuronas de la capa oculta.
  - Si M aumenta, la dimensión del espacio al cual se transforman los datos aumenta y por tanto la complejidad del modelo.
  - Si M disminuye, la dimensión del espacio al cual se transforman los datos disminuye y por tanto la complejidad del perceptrón.
- Si la función de activación de las neuronas es sigmoide la salida se puede interpretar como probabilidad y se aplica a clasificación. En caso contrario se aplica a predicción.

Perceptrón multicapa (VI)

Propiedades, limitaciones

- Algoritmo de entrenamiento sencillo que trabaja “on line”
- Es capaz de aproximar funciones no lineales continuas con el error deseado
- El método de optimización es propenso a caer en mínimos locales. El parámetro η no es fácil de fijar
- Sufre el problema de olvido catastrófico. Implementa funciones aproximadoras definidas globalmente
- La representación del perceptrón es distribuida y no es fácil interpretarla en forma de reglas
- Es difícil analizar la complejidad del perceptrón

Perceptrón multicapa (VII)

Extensiones, generalizaciones

- Controlar la complejidad mediante un término de regularización:

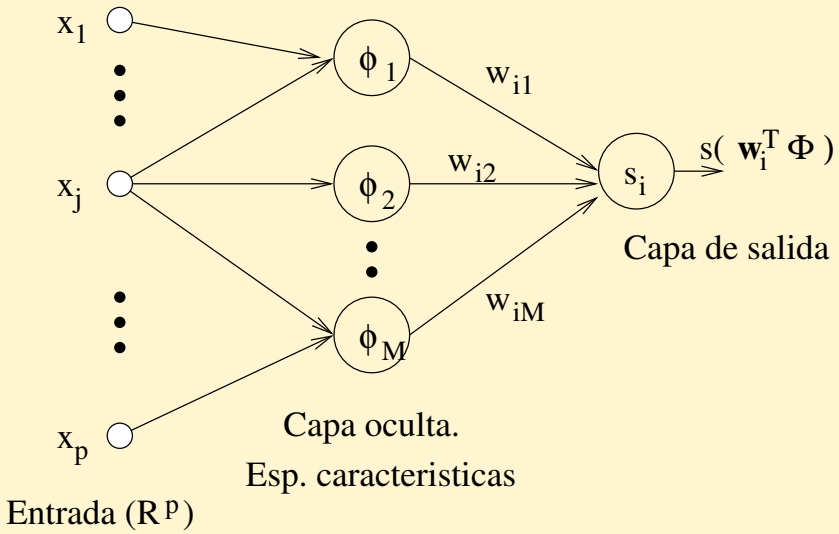
$$\Omega(\boldsymbol{w}) = E_D(\boldsymbol{w}) + \frac{1}{2}\sum_{ij} w_{ij}^2 \tag{47}$$

- Optimizar la verosimilitud (32) está más justificado en clasificación que el error cuadrático medio. La función de error (32) se conoce como entropía cruzada.

- Se puede optimizar el error cuadrático utilizando técnicas basadas en el Hessiano como el método de Newton.

Redes RBF (I)

Arquitectura



- Las neuronas de la capa de salida implementan la siguiente función:

$$o_k(\boldsymbol{x}) = \sum_{j=1}^{c} w_{kj}\phi_j(\boldsymbol{x}) + w_{k0} \tag{48}$$

La capa de salida realiza un análisis discriminante lineal en el espacio generado por las neuronas de la capa oculta.

- La salida de las neuronas de la capa oculta vale:

$$\phi_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{49}$$

La capa oculta transforma no linealmente los datos para convertirlos en lineales.

Redes RBF (II)

Error a optimizar

$$E_p(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{c}\{o_k(\boldsymbol{x}^p) - t_k^p\}^2 \tag{50}$$

Algoritmo aprendizaje

Se optimiza la función de error en dos pasos:

- Se suponen conocidos los parámetros de la capa oculta $\{\sigma_j, \mu_j\}$ y se optimizan los pesos de la capa de salida w_ij.

Como la capa de salida realiza un análisis discriminante lineal en el espacio generado por las ϕ_j la matriz de pesos se puede obtener aplicando la ecuación (23)

$$\mathbf{W}^T = \mathbf{\Phi}^\dagger \boldsymbol{T}\,, \tag{51}$$

donde

$$(\mathbf{T})_{pk} = t_k^p \qquad (\mathbf{\Phi})_{pj} = \phi_j(\boldsymbol{x}_p) \qquad (\mathbf{W})_{kj} = (w_{kj}) \tag{52}$$

- Se suponen conocidos los pesos de la capa de salida w_{ij} y se calculan los parámetros de la capa oculta $\{\mu_j, \sigma_j\}$

Métodos para obtener μ_j

- Muestreando aleatoriamente sin reemplazo. No óptimo.
- Algoritmo de cluster k-medias: Busca una partición en M-grupos $\{S_1, \ldots, S_M\}$ que minimice la suma de los las distancias al cuadrado dentro de cada grupo:

$$J = \sum_j \sum_{x \in S_j} \|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2 \tag{53}$$

  donde $\boldsymbol{\mu}_j = \frac{1}{N} \sum_{x \in S_j} \boldsymbol{x}$ son los centros de cada grupo.
- Modelos de Mixturas

Los algoritmos anteriores no consideran las etiquetas, no son óptimos.

Métodos supervisados para fijar los centros

- Método hacia adelante:
  - Partimos de un único patrón como centro, que será el que minimice el error de test.
  - En cada iteración añadimos un nuevo centro, que será el patrón que reduzca más el error de test.
- Método hacia atrás:
  - Partimos de todos los patrones como centros de la red.
  - Iterativamente elinamos el centro que contribuye a reducir menos el error sobre el conjunto de test.

Determinación de σ_j

Redes RBF (III)

Aspectos prácticos

- Determinación del número de neuronas de la capa oculta. Complejidad óptima del modelo.
- Función de activación para las neuronas de la capa de salida. Sigmoide para clasificación y lineal para predicción.
- Término de regularización en la función de error. Obtención parámetro regularización, complejidad óptima.

Redes RBF (IV)

Propiedades

- Aproximador universal.
- Algoritmo de optimización eficiente que se puede resolver con operaciones de álgebra lineal.
- No sufre el problema de “olvido catastrófico”.
- Funciones de activación de la capa oculta locales. Se puede interpretar en forma de reglas.

Tópicos avanzados en PR

- Tratamiento de datos no vectoriales (no estándar)
- Combinación de diferentes modelos
- Interpretación de las redes neuronales en forma de reglas
- Aprendizaje semi-supervisado con pequeños porcentajes de patrones etiquetados
- Aprendizaje por recompensa. No supervisado/supervisado
- Aprendizaje con costes asimétricos
- Identificación de clases “raras”