Clasificación (LVQ)

Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid)

Métodos de Clasificación: LVQ

- **Redes de cuantización vectorial (*Learning Vector Quantization*)**
  - Versión supervisada del método de Kohonen
  - Se conoce de antemano el número de clases (ejemplos etiquetados)
    - El número de células de la capa F2 puede ser mayor que el número de clases (varios prototipos por clase)
  - Clasificación supervisada por vecindad
  - Inicialmente los prototipos se distribuyen de forma aleatoria por el espacio de entrada. Una vez entrenada la red se obtendrá la solución al problema de clasificación:
    - Colocación final de los prototipos
    - Regla del vecino más cercano
  - Para clasificar un patrón de test:
    - Se introduce el nuevo ejemplo a clasificar
    - Se calcula su distancia a todos los prototipos existentes
    - Se etiqueta el nuevo ejemplo con la clase del prototipo más cercano

LVQ

- **Entrenamiento**
  - Se distribuyen los prototipos de forma aleatoria por el espacio de entrada
    - Pueden asignarse el mismo número de prototipos por clase o un valor proporcional al número de ejemplos de cada clase.
  - A medida que se van introduciendo los ejemplos de entrenamiento se van desplazando los prototipos
  - Misma regla que en método de kohonen:

$$\frac{d\mu_{ij}}{dt} = \alpha(t)\tau_j(t)(e_i(t) - \mu_{ij}(t))$$

  - Pero en este caso, τ_j varía de la siguiente manera:

$$\tau_j = \begin{cases} 1 & \text{si } C_j \text{ es ganadora y pertenece a la misma clase que } e_i \\ -1 & \text{si } C_j \text{ es ganadora y pertenece a distinta clase que } e_i \\ 0 & \text{si } C_j \text{ no es ganadora} \end{cases}$$

Comparación LVQ vs SOM

- Al ser supervisado se ha eliminado el concepto de vecindario
  - Sólo se modificará la célula ganadora
  - En SOM se modifica todo el vecindario
- La modificación de la célula ganadora no siempre será en la misma dirección
  - Si misma clase, la célula se aproxima al dato de entrada (se refuerza la salida de la red)
  - Si clase distinta, la célula se aleja (se penaliza)
  - En SOM siempre se acerca

Comparación LVQ vs SOM

- La tasa de aprendizaje también se decrementa con el tiempo

- Los prototipos se van acercando paulatinamente a los ejemplos cuya clase representan y se van alejando de los ejemplos de la clase contraria

LVQ-pak

- The Learning Vector Quantization Program Package
- Public-domain software package
- Download: http://www.cis.hut.fi/research/lvq_pak/
- Data file formats:

```
3
# First the yellow entries
181.0   196.0    17.0   yellow
251.0   217.0    49.0   yellow
# Then the red entries
248.0   119.0   110.0   red
213.0    64.0    87.0   red
```

núm de dimensiones

comentarios

Cada fila un dato, al final la etiqueta

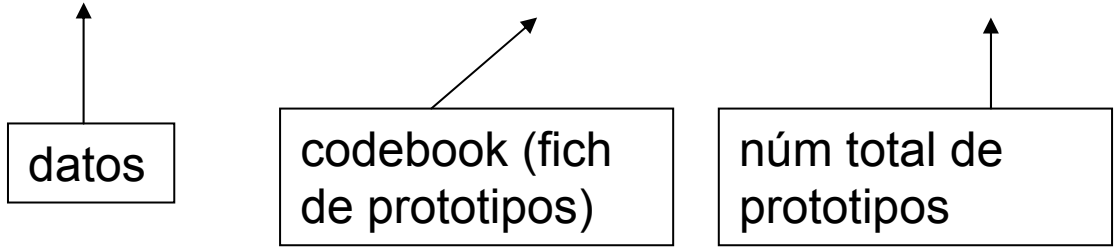
LVQ-pak. Parámetros de los programas

-noc	Number of codebook vectors in the codebook.
-din	Name of the input data file.
-dout	Name of the output data file.
-cin	Name of the file from which the codebook vectors are read.
-cout	Name of the file to which the codebook vectors are stored.
-rlen	Running length (number of steps) in training.
-alpha	Initial learning rate parameter.
-epsilon	Relative learning rate parameter (needed in the *lvq3* program).
-win	Window width parameter (needed in the *lvq2* and *lvq3* programs).
-knn	Number of neighbors used in knn-classification.
-alpha_type	The learning rate function type (in training routines). Possible choices are linear function (*linear*, the default) and inverse function (*inverse_t*). The linear function is defined as $\alpha(t) = \alpha(0)(1.0 - t/rlen)$ and the inverse function as $\alpha(t) = C\alpha(0)/(C + t)$ to compute $\alpha(t)$ for an iteration step t. In the package the constant C is defined to be $C = rlen/100.0$.
-version	Gives the version number of LVQ_PAK.

Uso de los programas

- **Primer paso: Inicialización de los prototipos (codebook)**
  - eveninit: crea el mismo número de prototipos por clase
  - propinit: número de prototipos proporcional al número de ejemplos de cada clase

```
eveninit -din diabetesTrain.txt -cin diabetes.cod -noc 100
```



Uso de los programas

- **Primer paso (adicional). Ajuste de los prototipos**
  - Se puede comprobar el número de prototipos por clase y la mediana de las distancias más cortas entre prototipos con mindist:

```
mindist -cin diabetes.cod
```

    - Salida:

```
In class          0   64 units, min dist.:   0.231
In class          1   36 units, min dist.:   0.331
```

  - Se puede ajustar la distribución inicial de los prototipos con balance (ver documentación)

```
balance -din diabetesTrain -cin diabetes.cod -cout diabetes1.cod
```

Uso de los programas

- **Segundo paso: Entrenamiento**
  - Existen algunas variantes del algoritmo básico LVQ. El más robusto y rápido es olvq1(optimized learning-rate LVQ1)

```
olvq1 -din diabetesTrain -cin diabetes1.cod -cout diabetes2.cod -rlen 5000
```

  - Ahora el 'codebook' de salida contiene los prototipos entrenados (diabetes2.cod)

Uso de los programas

- **Tercer paso: Evaluación de la capacidad de clasificación con un fichero de test**
  - El programa **accuracy** permite comprobar la precisión de la clasificación del mapa de prototipos con un fichero de datos independiente del utilizado para el entrenamiento:

```
accuracy -din diabetesTest -cin diabetes2.cod
```

- **Salida:**

Recognition accuracy:

0: 127 entries 84.25 %
1: 65 entries 63.08 %

Total accuracy: 192 entries 77.08 %

Uso de los programas

- **Tercer paso (continuación): Clasificación individual de los patrones de test**
  - El programa classify permite clasificar los vectores cuya clase se desconoce (patrones de test). El resultado lo guarda en un fichero.

```
classify -din diabetesTest -cin diabetes2.cod -dout diabetesTest.txt
```

fichero de salida con los datos clasificados

Uso de los programas

Fichero de salida:

0.411765	0.9799	0.57377	0.333333	0.171395	0.374069	0.0362938	0.566667	1
0	0.909548	0.721311	0.444444	0.602837	0.645306	0.0614859	0.0833333	1
0.0588235	0.643216	0.393443	0.454545	0.229314	0.603577	0.228437	0.05	**0**
0.0588235	0.758794	0.491803	0	0	0.388972	0.0431255	0.0166667	0

Fichero original de test:

0.411765	0.9799	0.57377	0.333333	0.171395	0.374069	0.0362938	0.566667	1
0	0.909548	0.721311	0.444444	0.602837	0.645306	0.0614859	0.0833333	1
0.0588235	0.643216	0.393443	0.454545	0.229314	0.603577	0.228437	0.05	1
0.0588235	0.758794	0.491803	0	0	0.388972	0.0431255	0.0166667	0

Uso de los programas

- **Cuarto paso (opcional): Visualización del codebook**
  - El programa sammon permite realizar una proyección del espacio n-dimensional a un plano aproximando las distancias euclídeas de los prototipos en el espacio n-dimensional.
    - Genera un fichero de texto con las coordenadas 2D de los prototipos y si se da la opción –ps genera un postcript con la imagen

```
sammon -cin diabetes2.cod -cout diabetes.sam -ps
```