Práctica de redes competitivas

Vamos a utilizar redes competitivas con topología para nuestro ejemplo de datos sobre zonas residenciales.

1. Tratamiento mínimo

Nos planteamos un agrupamiento de los datos residenciales, imponiendo una cierta topología entre las clases. Vamos a proponer 6 clases; como son pocas no merece la pena probar dimensiones superiores a 2 y vamos a ponerlas colocadas según una plantilla hexagonal.

1. Cargamos los datos
2. Creamos la red con `redt=selforgmap([2 3]);` donde la plantilla hexagonal es por defecto y hemos elegido un patrón de 2 por 3
3. Hacemos 50 pasadas como mínimo, pudiendo plantearnos hasta 200, con los comandos: `redt.trainParam.epochs=50;redt=train(redt,p);`
   y vemos los centroides con `redt.IW{1}`. Cada fila se corresponde con una categoría y cada columna con una variable. Expresa resumidamente las diferencias entre clases.

2. Afinado de ajuste

Ahora vamos a tener en cuenta los datos de salida en el agrupamiento y a elegir con más cuidado los parámetros que hemos cogido antes de manera irreflexiva.

1. Escalado de los datos. Antes no hemos hecho: los datos se han manejado como estaban en el fichero. Si una variable tenía un rango entre 1 y 1000 y otra entre 0 y 2, cuando se calcule la distancia entre dos vectores de esas componentes, ¿pesarán lo mismo? ¿Es esto lo que queremos? Las posibilidades que ofrece Matlab son (podemos aplicarlas por separado a distintas variables):

a) mapminmax: normalización a rango común predefinido

b) mapstd: normalización a varianza 1

2. Variables elegidas. Hemos cogido las variables elegidas sin más. ¿Necesitamos todas? ¿Son directamente esas las que dan mejor agrupamiento? ¿Y la variable de salida? Las posibilidades fáciles en Matlab son:

   a) Concatenar variables. Si una variable aparece varias veces, es una manera trucada de aumentar su importancia.

   b) processpca para elegir combinaciones de variables significativas por el método de componentes principales

3. Dimensión de la topología. Es generalizado coger dos dimensiones, pero ¿estamos seguros que es lo mejor para nuestros datos? Como norma general dimensiones mayores posibilitan mayores interrelaciones entre grupos. Pero las herramientas gráficas disponibles sólo funcionan bien en dos dimensiones.

4. Cantidad de grupos. ¿Por qué dos por tres? Existen dos posibilidades sensatas:

   a) Poner una cantidad claramente elevada de grupos y luego reagruparlos

   b) Hacer un agrupamiento previo que nos dé una idea

5. Topología: ¿por qué hexagonal? Podemos analizar el solapamiento de un agrupamiento previo con el comando silhouette. Posibilidades que ofrece Matlab:

   a) randtop por probar a no imponer nada

   b) gridtop en casos de bajo solape entre grupos (si hemos hecho un agrupamiento previo)

   c) hextop para más solape

6. Comprobación: sea cuál sea la combinación que probemos, hay medidas que nos pueden indicar si la cosa va bien. A continuación se comentan algunas, aunque no son directas de obtener:

   a) El error de topología: fracción de puntos cuyo segundo mejor no es adyacente. Idealmente debería ser 0

   b) Error de cuantización: la distancia media entre cada ítem y su «representante» Idealmente debería ser baja.

   c) Correlación entre la distancia entre puntos y la distancia entre sus «representantes» Idealmente debería ser 1

7. Los grupos, ¿qué representan? Podemos mirar el gráfico de planos y seleccionar aquellos pesos donde la diferencia sea mayor.

2.1. Caso ejemplo

Vamos a tomar varias decisiones de entre lo comentado anteriormente:

- Vamos a normalizar las variables de entrada a varianza unidad. La de salida no la normalizaremos.
- Elegiremos por componentes principales las combinaciones de variables de entrada que expliquen el 80 % de la varianza. A estas les añadiremos la de salida.
- Haremos un tanteo con el agrupamiento jerárquico. De él sacaremos una idea de cantidad de grupos y solape entre ellos (analizado con silhouette), lo que nos llevará a organizarlos en cierta dimensión y topología. Nos quedaremos en dos dimensiones.
- Tras ajustar la red comprobaremos la calidad de su agrupamiento con la «silueta»
- Probaremos la red con algún valor para ver qué pasa
- Crearemos una red sólo para entradas que podremos usar para hacer predicciones. Probaremos alguna.

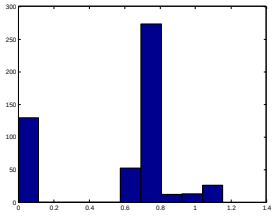
Veamos los comandos:

1. Cargar los datos, escalarlos y reducirlos a sus componentes principales, añadiendo la variable de salida

```
load housing;
[pesc,datesc]=mapstd(p); %entradas a varianza 1
[pimp,datpca]=processpca(pesc,0.2);
datos=[pimp;t];
```

2. Probar un agrupamiento jerárquico para estimar una cantidad de clases. Miramos la inconsistencia de la distancia de los nodos de agrupamiento. El coeficiente de inconsistencia es la diferencia respecto a la media (de los inferiores) dividido por la desviación típica.

```
arbol=linkage(datos');%Tantear jerárquico
inc=inconsistent(arbol);%Análisis del árbol
vecinc=inc(:,4);%La cuarta columna
hist(vecinc)
```



Viendo lo anterior, podemos quizá admitir hasta 0.9 ó 1 ó 1.1 Supongamos que nos decidimos por ser más exhaustivos y cogemos 1.1 Ahora indicamos que queremos un agrupamiento hasta una inconsistencia de 1.1

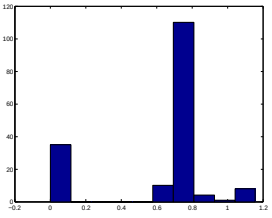
```
grupos=clusterdata(datos',1.1);
max(grupos)% Salen 173
```

3. Crear una red de agrupamiento, con un número de clases similar (por ejemplo, 13 por 13 dan 169):

```
redtot=selforgmap([13 13]);
redtot=train(redtot,datos);
pesostot=redtot.IW{1};
```

4. Como 169 grupos pueden ser muchos, vamos a agrupar las clases, creando «superclases» Para este agrupamiento, las entradas son los centroides obtenidos en el paso anterior. Podemos hacer con ellos un agrupamiento jerárquico.

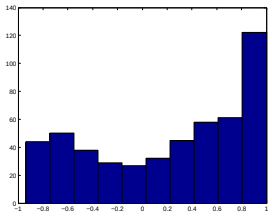
```
arbolpesos=linkage(pesostot);%vamos a reagruparlos
incarbol=inconsistent(arbolpesos);
hist(incarbol(:,4));
```



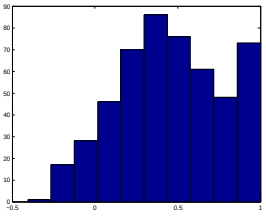
```
grupesos=clusterdata(pesostot,1.1);
```

5. Vamos a comparar el agrupamiento jerárquico, el dado por las clases originales de la red y el de las superclases obtenidas por agrupamiento de éstas. Utilizaremos la «silueta» que nos compara, para cada punto, la distancia media a puntos de su mismo grupo con la distancia media a puntos de los otros grupos, poniéndolo en escala [-1,1]:

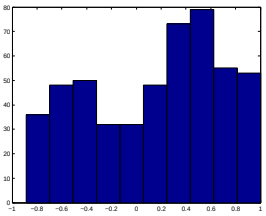
```
sil=silhouette(datos',grupos);
hist(sil)
```



```
grupred=redtot(datos);
[ngrupred,d]=find(grupred);
silred=silhouette(datos',ngrupred);
hist(silred)
```



```
agrupred=grupesos(ngrupred);
silrred=silhouette(datos',agrupred);
hist(silrred)
```



Se considera mejor aquellos en que hay más valores cerca de 1 (más alto indica más diferencia entre la distancia con puntos de otros grupos que con puntos de su propio grupo)

6. Vamos a probar un punto y a ver qué tal es representado por el centroide correspondiente elegido por la red, por lo menos en su variable de salida:

```
%Probamos con, por ejemplo, el punto 250
p250=p(:,250);
pesc250=mapstd('apply',p250,datesc);
pimp250=processpca('apply',pesc250,datpca);
dat250=[pimp250;t(250)];
grupored250=redtot(dat250);
find(grupored250==1);%Sale 145
grupesos(grupored250==1)%Sale la superclase 25
find(grupesos==25)%Sale el 145 y los que están con él en la superclas
pesostot(145,14)%Sale 26.4
t(250)%Sale 26.2
```