APRENDIZAJE BASADO EN CASOS

Índice

- Introducción
- K-Nearest Neighbor
- Regresión local ponderada
- Razonamiento basado en casos
- Algoritmos perezosos vs. algoritmos ansiosos

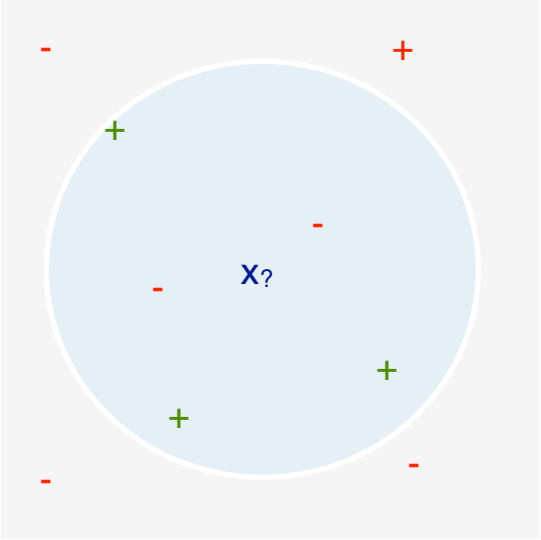
Introducción

- La idea es crear un clasificador ‘perezoso’.
- Para clasificar una nueva instancia, utilizo aquellas que más se le parecen de las que ya conozco.
- Ventajas:
  - Para cada nueva instancia puedo obtener un clasificador diferente.
  - La descripción de las instancia puede ser tan compleja como quiera.
- Desventajas:
  - El costo de clasificación puede ser alto.
  - Atributos irrelevantes pueden afectar la medida de similitud.

K - Nearest Neighbor

- Queremos aproximar un concepto, utilizando las k instancias más cercanas a un elemento <a_1,...,a_n> que deseamos clasificar.
- Utilizamos la función de distancia euclidiana:

$$d(< a_1, \ldots, a_n >, < b_1, \ldots, b_n >) = \sqrt{\sum (a_i - b_i)^2}$$



- El valor que damos para $x_?$ es:

$$h(x_?) \leftarrow argmax_{v \in +,-} \sum_{x_i \in k-nn(x_?)} \delta(v, f(x_i))$$

- Si la función es real, se puede utilizar:

$$h(x_?) \leftarrow \frac{1}{k} \sum_{x_i \in k-nn(x_?)} f(x_i)$$

K - Nearest Neighbor

- Los ejemplos más cercanos a la instancia $x_?$ podrían considerarse más importantes que los lejanos...

$$h_{dis}(x_?) \leftarrow argmax_{v \in \{+,-\}} \sum_{xi \in k-nn(x_?)} w_i \,.\, \delta(v, f(x_i))$$

$$h_{real}(x_?) \leftarrow \sum_{x_i \in k-nn(x_?)} \frac{w_i \,.\, f(x_i)}{\sum w_i}$$

en donde: $w_i = \frac{1}{d(x_i, x_?)^2}$

- Si ponderamos por la distancia, podemos usar a todo el conjunto para evaluar una nueva instancia. [Método de Shepard]

K - Nearest Neighbor

- Al tomar en cuenta a varios vecinos, el algoritmo es menos sensible al ruido.
- Pero ¿cuántos vecinos elegir?
  - Si se elige k muy bajo, el resultado es muy sensible al ruido; si es muy alto, las zonas que tengan muchos ejemplos pueden acaparar a zonas que tengan menos.
  - Por lo general se elige un k impar para no tener problemas de empate. Los valores usuales son bajos: 1, 3 y 5.
  - Una forma de estimar k es probando distintos valores, midiendo la performance dejando un elemento del conjunto afuera y clasificando con el resto (1-out-cross-validation).

K - Nearest Neighbor

- A diferencia de los árboles de decisión, al aplicar la distancia euclidiana, se tienen en cuenta TODAS las dimensiones que describen un atributo

- Problemas:
  - si 2 atributos en 20 son los relevantes, instancias que en realidad son muy diferentes pueden estar muy próximas en el espacio (maldición de las dimensiones).

  - ¿Qué sucede con atributos con distintas magnitudes?

K - Nearest Neighbor

- Maldición de las dimensiones:
  - Una posible solución es asignar pesos a los distintos atributos. Esto corresponde a modificar el largo de los ejes del espacio.
  - ¿Cómo se determina cuánto hay que acortar o alargar un eje?
    - Penalizando los atributos cuyos distribución es uniforme en cada clase.
    - Utilizando validación cruzada.

K - Nearest Neighbor

- ¿Qué sucede con atributos con distintas magnitudes?
  - Estandarización: se asume que A es $\mathcal{N}(\mu_A,\sigma_A)$ y se lleva a $\mathcal{N}(0,1)$

$$Est(x, A) = \frac{x - \mu_A}{\sigma_A} \qquad \text{(z score)}$$

  - Reescalamiento: se cambia el rango de los atributos

$$Resc(x, A) = \frac{x - min_A}{max_A - min_A} \qquad RescAbs(x, A) = \frac{x}{max_A - min_A}$$

  - *OneHot: se* transforma atributos categoriales en vectores

$$OneHot(x_2, \{a_1, a_2, \ldots, a_n\}) = (0,1,\ldots,0)$$

K - Nearest Neighbor

- Cada clasificación implica la selección de k ejemplos; esto puede ser muy costoso si contamos con un gran conjunto de ejemplos.

- Esto se soluciona implementando una buena indexación sobre el conjunto.

- ¿Cuál es el sesgo inductivo de K-NN?

  La clasificación de una instancia es parecida a las de sus k vecinos (cercanía implica similitud).

Regresión local ponderada

- Generalizamos KNN, creando una aproximación local de la función objetivo y construimos una función h que aproxime a los puntos cercanos a $x_?$
- Podemos elegir qué modelo de función utilizamos para esta aproximación (lineal, cuadrática, etc.).
- Por ejemplo:

$$h(< a_1, \ldots, a_n >) = w_0 + w_1 . a_1 + \ldots + w_n . a_n$$

  - Buscamos al vector $(w_0, w_1, \ldots, w_n)$ que minimice el error de la función sobre los k puntos cercanos.
  - Pero, ¿qué función de error deberíamos considerar?

Regresión local ponderada

- Algunas opciones:
  - Sobre los k vecinos cercanos:

$$E = \frac{1}{2} \sum_{x \in k-near(x_?)} (f(x) - h(x))^2$$

  - Ponderando a todo el conjunto:

$$E = \frac{1}{2} \sum_{x \in D} (f(x) - h(x))^2 K(d(x, x_?))$$

  - Ponderando a los k-cercanos:

$$E = \frac{1}{2} \sum_{x \in k-near(x_?)} (f(x) - h(x))^2 K(d(x, x_?))$$

Donde K es una función decreciente.

Regresión local ponderada

- En la selección de la función de error se debe considerar cuánto queremos que influya el total del conjunto de entrenamiento vs. el costo computacional.

- Debido al costo en la reducción del error, se utilizan por lo general funciones lineales o cuadráticas.

- Luego de clasificar la nueva instancia, podemos descartar a la hipótesis encontrada: cada instancia genera una nueva aproximación.

Razonamiento basado en casos

- ¿Qué sucede cuando las instancias son representadas de forma más compleja?

Transporte: <ómnibus>
Tiempo: <dia_{+1}: nublado, dia_{+2}: soleado, dia_{+3}:?>
Lugar: <casa[cuartos: 2, piscina: no), centro:10 km>
Personas:<adultos(hombres: 1, mujeres: 1), niños=4>

} Satisfacción: ????

- Al igual que KNN y RLP, clasificamos una instancia en base a casos parecidos: en lugar de puntos en un espacio euclídeo, las instancias se representan con atributos más complejos.
- Se debe buscar una métrica de similitud que depende del dominio de trabajo.
- La solución se basa en combinaciones complejas y específicas al dominio de aplicación.

Algoritmos perezosos vs. ansiosos

- Los algoritmos que vimos (KNN, Regresión Local…) difieren el cálculo de una hipótesis hasta la llegada de una nueva consulta (algoritmos perezosos).
- Estos algoritmos computan una aproximación local de la función objetivo para responder cada nueva consulta: utilizan múltiples aproximaciones locales para modelar la función objetivo [global].
- Los algoritmos ansiosos pueden utilizar también aproximaciones; sin embargo, éstas quedan «fijas» al conjunto de entrenamiento.

Algoritmos perezosos vs. ansiosos

- Dado un mismo espacio de hipótesis, los algoritmos perezosos tienen una mayor poder de adaptación a una nueva consulta.
- El costo de clasificación, sin embargo, aumenta: la aproximación se realiza en tiempo de clasificación y no de entrenamiento.
- Se precisan estructuras eficientes para determinar los ejemplos cercanos a la instancia a clasificar.