Apuntes Tema 1

Albert Ribes Marzá

5 de octubre de 2017

Resumen

Los apuntes que vaya tomando en clase

1.

1.1. Introducción a ML

Ejemplo 1 Se pretende medir la temperatura (t) en un punto de una central nuclear, pero la temperatura es tan alta que no se puede medir directamente con ningún sensor. Se intentará deducir la temperatura así:

- t - temperatura a predecir (variable)
- x - vector de variables medibles que posiblemente inciden en t
- z - vector de variables **NO medibles** que posiblemente inciden en t

La relación completa es $t = \delta(x, z)$, que es una función.

Pero no conocemos z ➔Aun conociendo x, el valor de t oscila. La relación entre t y x se hace *estocástica*

$p(x, t)$ será la probabilidad de que con esa x se tenga esa temperatura.

Hay que construir una función $t = y(x)$ donde t sea el valor más plausible. El problema es que no conocemos p

La forma de atacar el problema será recolectar datos $\{(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)\} = D$

$(x_n, t_n) \sim^{i.i.p.} p$ (variables independientes e identicamente distribuidas)

El objetivo del *Machine Learning* (ML) es obtener y a partir de D. En este caso es un problema de *regresión*

Ejemplo 2 Se tiene una planta de reciclaje, y se quieren clasificar los objetos que van pasando por la cinta. Los datos son:

- t - tipo de producto
- x - los atributos de los productos que captamos con una cámara
- z - los atributos que no captamos con la cámara

Es el mismo problema de antes, pero ahora t es discreta. Se trata entonces de un problema de *clasificación*.

1.2. Ejemplo introductorio: el ajuste polinómico

Tenemos $x \in \mathbb{R}$ y queremos predecir $t \in \mathbb{R}$ (*Regresión*)
En todos los problemas nos encontraremos con:

- $x_n \in (0,1)$, $x_n \sim U(0,1)$, (un conjunto de observaciones)
- $t_n = sin(2\pi x_n) + \varepsilon, \varepsilon \in N(0,\sigma^2)$, normalmente $\sigma^2 = 0{,}3^2$, y donde ε es el ruido aparentemente aleatorio, que proviene de los datos que no conocemos.

Vamos a intentar ajustar los datos. Sabemos que si los datos son continuos (no dan saltos) podemos ajustarlos con un polinomio en un intervalo:

- $P_n := \{c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n\} = \{\sum_{i=0}^{n} c_i x^i | c_i \in \mathbb{R}\}$
- $C \in \mathbb{R}^{n+1}$ son todos los parámetros.
- Llamamos $y(x;c) = \sum_{i=0}^{M} c_i x^i$ un modelo
- Respecto a X, y es una función no lineal
- Respecto a C, y es una función lineal

Diremos que un modelo es lineal cuando lo es respecto a los parámetros.
Ajustamos y a los datos $\{(x_1,t_1),(x_2,t_2),\ldots(x_n,t_n)\}$ definiendo una función de error (la función de error cuadrático):

$$E(c) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n;C) - t_n)^2$$

Como E depende automáticamente de C, derivamos e igualamos a 0 para encontrar el mínimo. Hay que hacer n derivadas, una para cada c_j:

$$\frac{\partial E}{\partial c_j} = \frac{1}{2}\sum_{n=1}^{N} 2(y(x_n;C) - t_n)(x_n)^j$$

$$\frac{\partial E}{\partial c_j} = \sum_{n=0}^{N}\Big(\sum_{i=0}^{M}(c_i x_n^i - t_n)\Big)x_n^j = 0$$

El problema de todo esto es que no sabemos qué grado de polinomio deberíamos usar para reflejar el comportamiento de la variable.

- Si es demasiado pequeño no seremos capaces de ajustar la parte regular (y) de los datos (infra-ajuste)
- Si es demasiado grande se ajustará la parte regular (y) y también el ruido (sobre-ajuste)

Cómo elegir el grado del polinomio?

Únicamente conociendo $E(C)$ no se puede saber. Para hacerlo se usa una muestra alternativa de datos de validación. Esta muestra debería tener más datos.

Si observamos el error producido en nuestros datos con diferentes grados de polinomios, obviamente será más pequeño cuanto más grande sea el polinomio, puesto que tiene más flexibilidad. Pero si miramos qué ocurre con los datos de validación, veremos que al principio el error desciende, pero llega un punto en que empieza a subir. El punto mínimo se corresponde con el grado correcto.

El error empezará a subir porque el sobre-ajuste se ha adaptado a los datos aleatorios, pero en la muestra de validación no tienen por qué ser los mismos, y produce más error.

Si la muestra de validación tiene pocos datos, el mínimo estará poco definido, será más redondeado y más difícil de localizar.

Alternativa

Pero no siempre es posible tener suficientes datos de validación. Para esto hay una alternativa.

Uno se pregunta: ¿Si un polinomio de grado 9 "contiene" a los de grado más pequeño, no podría ocurrir que eligiéramos uno de grado mayor, y que él mismo anulara los coeficientes sobrantes hasta que sea del grado adecuado?

La respuesta es que espontaneamente esto no pasa, puesto que para igualar los datos aleatorios son necesarios coeficientes muy grandes. Si queremos que ocurra tenemos que forzarlo de alguna manera. Para hacerlo, redefiniremos nuestra función de error, de manera que también penalice los coeficientes demasiado grandes. Penalizaremos la norma 2, que equivale a la "distancia" pitagórica.

¿Pero cuanto tenemos que penalizarlo? Si nos pasamos o nos quedamos cortos no servirá de nada. Como no sabemos cuanto tenemos que penalizar, usaremos un parámetro λ que regulará la penalización que hacemos.

La función de error queda así:

$$E(C) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n;C) - t_n\Big)^2 \frac{\lambda}{2}||c||^2$$

El $\frac{\lambda}{2}$ es simplemente para que al derivar quede más simple. Podría ser solo λ

1.3. Conceptos de inferencia estadística

$D = \{x_1, \dots x_n\}$ es una realización de una variable aleatoria (v.a.) X_n que tiene una función de distribución $x_n;\theta), \theta \in \Theta$

El objetivo es obtener una estimación $\hat{\theta}$ de θ, dado D

La probabilidad de obtener D es:

$$P_n(D,\theta) = \prod_{n=1}^{N} p(x_n,\theta)$$

Definimos la función de verosimilitud (likelihood) así:

$$\mathcal{L} : \theta \to \mathbb{R}$$

$$\theta \to \mathcal{L}(\theta) = P_n(D;\theta)$$

El estimador de máxima verosimilitud es $\hat{\theta} = \text{argmax}\mathcal{L}(\theta)$, $\theta \in \Theta$

Si es de una sola variable, la forma de hacerlo es derivar e igualar a 0.

Es conveniente operar con el logaritmo de α, pues simplifica la maximización de un producto:

$$\ln(p_1 p_2 \dots p_n) = \ln(p_1) + \ln(p_2) + \dots + \ln(p_n)$$

Ejemplo

Por aquí en medio van más cosas, que todavia no he puesto

2. Reducción de dimensión

Para hacer *Machine Learning* es interesante tener los datos lo más simplificados posible, pues eso evita el sobre-ajuste. Existen muchos métodos para reducir la dimensión de un problema. Reducir la dimensión se podría entender como quedarse con una sombra de la imagen real que tenemos. Esto es: si todos los datos que tenemos estuvieran en 3 dimensiones, podría interesarnos trabajar con la sombra que proyectan esos datos, de manera que trabajaríamos con solo 2 dimensiones.

Pues hacemos los mismo, pero con muchas dimensiones.

Las ventajas que tiene esto son que evita el sobre-ajuste, nos permite entender mejor los datos y que son más fáciles de representar, con plots o dibujos.

Pero hay que cojer una buena proyección de los datos reales. Puesto que está claro que vamos perder información (datos, en realidad), cojeremos una proyección que refleje lo que nos interesa, y que deseche otras cosas.

Es por eso que hay muchas formas de reducir la dimensión de un conjunto de datos, cada una según la prioridad que uno tenga, y cogiendo las proyecciones más adecuadas para cada necesidad.

Ahora veremos algunas de las formas de reducir la dimensión:

2.1. Principal Components Analisis (PCA)

Este algoritmo tiene como prioridad preservar la varianza de los datos, maximizar la dispersión en las proyecciones.

Esto es, en la analogía de la sombra, cojer la sombra que tenga más area.

De forma más técnica:

Tenemos una muestra de datos $\{X_1, X_2, \dots, X_n\}, X_i \in \mathbb{R}^d$ que provienen de un vector aleatorio $X = \{X_1, \dots, X_n\}^T$. Cada una de las X_i es una variable (aleatoria?) y tenemos d muestras en cada una de las variables.

Disponemos también de la matriz de covarianzas Σ.

La matriz de covarianzas es una matriz de $n \times n$ donde Σ_{ij} es $\text{var}(X_i, X_j)$ si $i \neq j$ y Σ_{ii} es σ_i^2

Tenemos datos en n dimensiones, y decidimos que queremos únicamente k dimensiones, $k < n$, y no cualesquiera dimensiones, sino las que maximicen la varianza.

Hemos de encontrar entonces k vectores n-dimensionales. Encontraremos n vectores que serán todos "perpendiculares" entre ellos y cojeremos los k vectores que tengan más varianza.

Nuestro objetivo entonces obtener un nuevo sistema de coordenadas $Y = (Y_1, \ldots, Y_n)$ que cumpla estas condiciones:

1. $\text{Covar}(Y_i, Y_j) = 0$ si $i \neq j$
2. $\text{Var}(Y_1) > \text{Var}(Y_2) > \ldots > \text{Var}(Y_n)$ (de hecho los ordenaremos decrecientemente)
3. $\sum_{i=1}^{d} Var(X_i) = \sum_{i=1}^{d} Var(Y_i)$

Encontraremos la proyección Y_i encontrando un vector w_i que cumpla que $Y_i = w_i^T \cdot X$

Como hay muchos vectores que cumplen esa condición (vectores que tienen todos la misma dirección, pero distinto módulo) establecemos la condición sobre w_i de que la norma 1 sea 1, esto es: $||w_i|| = 1 \Rightarrow w_{i1}^2 + w_{i2}^2 + \cdots + w_{in}^2 = 1$

Objetivo: w_1 ha de maximizar la varianza de Y_i, sujeto a que $||w_i|| = 1$

$$Var(Y_i) = Var(w_i^T \cdot X) = w_1^T \cdot Var(X) \cdot w_i \tag{1}$$

Este último paso es algo que se sabe y que sale en Wikipedia. Nos lo creemos.

Para resolver un problema de maximización sujeto a algunas condiciones se hace con el método de los multiplicadores de Lagrange.

Anexo: método de Lagrange