Artificial Neural Networks - algoritmos

Martín Gutiérrez

August 7, 2022

Introducción

Cubrimos la base de las estructuras de ANN durante la clase pasada y sus contextos.

Durante esta sesión, vamos a cubrir los algoritmos que rigen el funcionamiento de las ANN.

Todo se reduce al error...

Se acuerdan cuando les pasé regresión lineal y les di como medida de calidad de una recta el error cuadrático medio?

$\rightarrow$ SSDN

Intuitivamente, el error es la guía

Hay muchas formas de cálculo para el error (error cuadrático medio es solo uno de ellos), pero se debe tener en cuenta que la dirección de aprendizaje de la ANN es guiada por el error. Lo hace de acuerdo a la siguiente expresión:

$$\theta \ := \theta - \alpha(y^{(i)} - \theta^T x^{(i)})x^{(i)}$$

Donde θ representa el vector de pesos (o de importancia de las sinápsis), α es la constante de aprendizaje, $y^{(i)}$ representa la respuesta correcta del i-ésimo ejemplo y $x^{(i)}$ son las entradas del i-ésimo ejemplo.

Ahora bien, qué representa esta expresión y de dónde viene?

Error cuadrático medio

Como ya vimos cuando hablamos de regresión lineal, una buena medida del error de una función que aproxima a otra es:

$$\frac{1}{2m}\sum_{i=1}^{m}(y^{(i)} - (\theta^T x^{(i)}))^2 \text{ o}$$

$$\frac{1}{2}(y^{(i)} - (\theta^T x^{(i)}))^2 \text{ , para un ejemplo particular}$$

Así pues, teniendo en cuenta que lo que entrega como salida una ANN es el resultado de aplicación de una función, una medida que puede servir para evaluar qué tan "bien" está ese valor es el ECM.

Súper, pero... qué hacemos con ese error?

La idea es que el error guíe la adaptación de la ANN hacia su aprendizaje.

En tal sentido, qué se hacía con esa función al hablar de regresión? Tiene alguna relación la ECM con la función descrita unas slides atrás que se emplea para la actualización de θ?

Se usa el error para ajustar la ANN

Como bien habrán deducido de la slide anterior, se debe minimizar ese error (y una forma en la que sabemos hacerlo utilizando descenso por el gradiente), puesto que es ese error el que mide el rendimiento de nuestra ANN.

El gradiente se calcula como $\nabla ECM(\theta)$ se calculará como el vector de los m, $\frac{\partial ECM}{\partial \theta_j} = (y_j^{(i)} - \theta^T x_j^{(i)}) x_j^{(i)}$, siendo x_j la j-ésima componente de $x^{(i)}$.

Ahora bien, esa medida hay que utilizarla para ajustar la ANN: la forma en la que se ajusta es que actúe sobre los parámetros libres de la ANN, - los pesos - (θ). Teniendo entonces el error obtenido al final del procesamiento hecho (hacia adelante), se utiliza el error en la otra dirección para el ajuste.

Regla Delta de entrenamiento

Por todo lo anterior, se obtiene el método de actualización que se denomina Regla Delta de Entrenamiento (para actualización de pesos) y viene dada por:

$$\theta := \theta - \alpha \nabla ECM(\theta)$$

o bien,

$$\theta_j := \theta_j - \alpha \sum_{k=1}^{n} (y_k^{(i)} - \theta^T x_k^{(i)}) x_k^{(i)}$$

Algoritmo de descenso por el gradiente

Así pues, se enuncia (y recuerda) el siguiente pseudo-código para el algoritmo de descenso por el gradiente (MLP):

DescensoGrad(*ejemplos*,α)

- Inicializar los θ_j en valores aleatorios pequeños.
- Repetir hasta que ocurra convergencia:
  - Inicializar $\Delta\theta_j = 0$
  - Para cada $< x^{(i)}, y^{(i)} > \in$ *ejemplos*:
    - Calcular la salida $\theta^T x$ de la unidad.
    - Para cada peso θ_j:

$$\Delta\theta_j := \Delta\theta_j + \alpha(y^{(i)} - \theta^T x^{(i)})x^{(i)}$$

  - Para cada θ_j:

$$\theta_j := \theta_j + \Delta\theta_j$$

Y en otros tipos de unidades?

Muchas veces, el perceptrón no es adecuado como unidad de base de las ANN. Así pues, una alternativa son las unidades *sigmoidales*.

Esto ya lo vimos cuando hablamos de regresión logística (la función del mismo nombre), e incluso calculamos la diferenciación.

Bajo tal escenario, se dan los siguientes cambios fundamentales:

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Por lo demás, es sabido que:

$$\frac{dg(x)}{dx} = g(x)(1 - g(x))$$

Proceso de entrenamiento

Habemos mostrado la forma de actualizar los pesos sobre los perceptrones y otro tipo de unidades usando el Error Cuadrático Medio. Según lo visto, si le agregamos a la unidad una función de "corte" que sea diferenciable, sólo cambia la expresión a evaluar según lo mostrado.

El proceso de actualización se aplica de igual manera con la redes multicapas, pero usando la regla sobre los pesos de manera estratificada y "en reversa": el gradiente asociado al error se propaga hacia atrás en la ANN.

A este proceso se le conoce como *Backpropagation* y ajusta los pesos en el proceso de aprendizaje. Será mostrado brevemente a continuación.

Backpropagation

Backpropagation supone que hay una red de neuronas sobre la que se está propagando el error. En este punto surgen nuevas variables:

$$o_1, \ldots, o_L$$

Representan las salidas efectivas de la red, y se usan con:

$$y_1, \ldots, y_L$$

Que representan los datos de entrenamiento de la red.

Entonces, tomando en cuenta los cálculos hechos previamente, sabemos que la última capa de la red absorbe el error directamente, por lo que definimos a continuación el algoritmo de *Backpropagation* para todas las capas.

Backpropagation

Se supone una red con capas escondidas y con conexiones completas en feed-forward con unidades sigmoidales:

BackPropagation($ejemplos, \alpha, n_{in}, n_{out}, n_{hidden}$)

- Inicializar todos los pesos en valores aleatorios pequeños.
- Repetir hasta que ocurra convergencia:
  Por cada ejemplo:
  - Procesar el ejemplo a través de la red y calcular la salida (los o_i)
  - Para cada unidad k de salida:
    $$\delta_k := o_k(1 - o_k)(y_k - o_k)$$
  - Para cada unidad escondida h:
    $$\delta_h := o_h(1 - o_h) \sum_{k \in \text{salidas}} \theta_{k,h} \delta_k$$
  - Se actualizan los pesos $\theta_{j,i}$:
    $$\theta_{j,i} := \theta_{j,i} + \alpha \delta_j x_{j,i}$$

Próxima clase

Habiendo entonces mostrado cómo funciona por debajo el entrenamiento de una ANN, están ya en condiciones de diseñar y operar una ANN básica de tipo Feed-forward. Durante la próxima clase nos enfocaremos en tecnologías correspondientes a las Deep Neural Networks (DNN) y cómo utilizar esas variaciones para el diseño de ese tipo de redes.