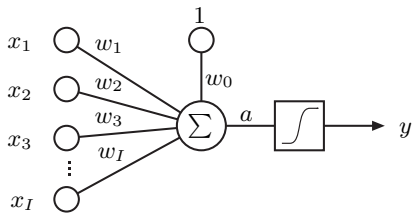
# CC50Q - Teoría de la Información y Redes Neuronales

Prof.: Pedro Ortega <peortega@dcc.uchile.cl>
Aux.: Francisco Claude <fclaude@dcc.uchile.cl>

30 de octubre de 2005

## 1. El perceptrón

El Perceptrón es el modelo matemático de una neurona (artificial) ilustrado a continuación.




Posee $I$ entradas $x_1, \ldots, x_I$ y produce un resultado $y$. Los valores $w_1, \ldots, w_I$, llamados *pesos*, escalan su entrada asociada. La neurona además posee una entrada especial $x_0 \equiv 1$, que es multiplicada por el *umbral de activación* $w_0$. Estos valores se suman luego en el combinador lineal "$\Sigma$". El resultado $a$ de esta operación es transformado a un valor cercano a uno (activación) o un valor cercano a cero por la función de activación sigmoidal (con forma de "S").

### 1.1. Evaluación del resultado

Dada una entrada $\mathbf{x} = [x_1, \ldots, x_I] \in \mathbb{R}^I$, el resultado de una neurona se evalúa como sigue.

1. *Combinación lineal*

$$a = \sum_{i=0}^{I} w_i x_i$$

2. *Activación*

$$y = f(a) = \frac{1}{1 + \exp -a}$$

La activación de la neurona puede verse como una división del espacio de entrada en dos mitades. El hiperplano dado por la ecuación

$$\sum_{i=0} w_i x_i = 0$$

constituye esta frontera, ya que los valores positivos producen resultados $y \approx 1$ y los valores negativos producen resultados $y \approx 0$.

### 1.2. Modularización de la estructura de la neurona

La neurona puede verse como la composición de dos módulos: el combinador lineal y la función de activación. Cada módulo puede verse como una función vectorial contínua que recibe una entrada y calcula una salida (que eventualmente puede alimentar módulos subsiguientes). Esta modularización es importante tenerla en mente para la derivación del algoritmo de entrenamiento.

### 1.3. Representación de funciones

Una neurona es capaz de representar una familia de funciones de la forma $f : \mathbb{R}^I \to [0; 1]$. En particular, adoptando una codificación apropiada (en la entrada y en la salida), se pueden representar funciones booleanas. Una codificación sencilla sería codificar $V \leftrightarrow 1$ y $F \leftrightarrow 0$. Cada elección de $\mathbf{w} \in \mathbb{R}^{I+1}$ da origen a una función diferente.

### 1.4. Calidad de la representación

Sea $g : \mathbb{R}^I \to [0;1]$ una función que se pretende representar por medio de un perceptrón. De esta función poseemos ejemplos de entrada-salida. Sea $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ el conjunto de entradas, y $D = \{d^{(1)}, \ldots, d^{(N)}\}$ el conjunto de salidas deseadas, de manera que $d^{(i)} = g(\mathbf{x}^{(i)})$ para toda entrada. Entonces, si la neurona produce salidas $y(\mathbf{x}^{(i)}; \mathbf{w}) \approx d^{(i)}$ se puede decir que la función $g$ queda bien representada por medio de la neurona.

Para formalizar esta noción de calidad de representación, se definen funciones de error. Las más importantes son

1. *Error cuadrático:*

$$E = \sum_{i=1}^{N} (d^{(i)} - y^{(i)})^2$$

   Mide el error absoluto incurrido en la representación. Sirve para medir la calidad de aproximación de una función continua.

2. *Entropía cruzada:*

$$E = -\sum_{i=1}^{N} d^{(i)} \ln y^{(i)} + (1 - d^{(i)}) \ln(1 - y^{(i)})$$

   Mide el error relativo incurrido en la representación, si la salidas $d^{(i)}$ y $y^{(i)}$ son interpretadas como probabilidades de una variable aleatoria binaria. Los valores de la salida deben estar en $[0;1]$ para que no se indefina este criterio.

## 2. Aprendizaje

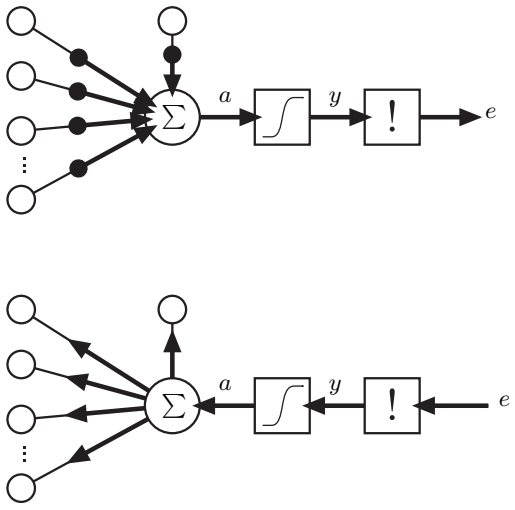
Dada una neurona Perceptrón y un conjunto de entrenamiento (i.e. el conjunto de ejemplos de entrada-salida), el aprendizaje de una neurona se reduce a un problema de optimización del criterio de error. Existen muchos algoritmos de optimización para este propósito, pero los más populares requieren del conocimiento del gradiente del función (con respecto a los pesos $\mathbf{w}$). En todos ellos, el objetivo es partir de una elección aleatoria de los pesos, y luego, usando la información sobre el gradiente, bajar por la superficie de error hasta dar con un mínimo. La regla de aprendizaje más sencilla es el *descenso por el gradiente*:

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \mu \frac{\partial E}{\partial \mathbf{w}}$$

donde $\partial E / \partial \mathbf{w}$ es el gradiente (interpretable como un vector en el espacio de los pesos) y el factor de aprendizaje $\mu$ que regula el tamaño de los pasos a dar en cada iteración.

### 2.1. Retropropagación del error

El gradiente del error podría calcularse en forma explícita, escribiendo la función de error completa y derivando según $\mathbf{w}$. Sin embargo, en la práctica se aplica un truco (usando la regla de la cadena) que resulta en un cálculo mucho más sencillo y modular. La idea consiste en calcular el error incurrido por la evaluación de un patrón, y a partir de este error, retropropagar una señal de error que irá construyendo el gradiente de a poco.







Entonces plantearemos dos algoritmos equivalentes de entrenamiento. Observar que el cálculo del gradiente consiste en hacer primero una evaluación hacia

adelante, y luego una evaluación hacia atrás, pasando por el mismo camino.

Queremos entrenar una neurona de $I$ entradas con una función de activación sigmoide logística

$$f(a) = \frac{1}{1 + \exp -a}$$

y con el criterio de error cuadrático

$$E = \sum_{i=1}^{N} (d^{(i)} - y^{(i)})^2.$$

Notemos que la neurona posee un total de $I + 1$ parámetros ajustables: los pesos y el umbral de activación.

ENTRENAR$(X, D, \mu, \epsilon) \rightarrow \mathbf{w}$:
*Inicializar* $\boldsymbol{w}$*:*
$\mathbf{w} \leftarrow$ VECTORALEATORIO$(I + 1)$
*Iterar hasta que* $E$ *sea inferior a* $\epsilon$*:*
Do {
  *Inicializar* $E$, $\partial E/\partial \boldsymbol{w}$*:*
  $E \leftarrow 0$, $\partial E/\partial \mathbf{w} \leftarrow \mathbf{0}$
  *Calcular* $E$, $\partial E/\partial \boldsymbol{w}$*:*
  For $i = 1, \ldots, N$\{
    *Calcular* $e^{(i)}$*:*
    $a^{(i)} \leftarrow \sum_{j=0}^{I} w_j x_j^{(i)}$
    $y^{(i)} \leftarrow 1/(1 + \exp -a^{(i)})$
    $e^{(i)} \leftarrow (d^{(i)} - y^{(i)})^2$
    $E \leftarrow E + e^{(i)}$
    *Calcular* $\partial e^{(i)}/\partial \boldsymbol{w}$*:*
    $\frac{\partial e^{(i)}}{\partial y^{(i)}} \leftarrow -2(d^{(i)} - y^{(i)})$
    $\frac{\partial y^{(i)}}{\partial a^{(i)}} \leftarrow y^{(i)}(1 - y^{(i)})$
    $\forall j, \frac{\partial a^{(i)}}{\partial w_j} \leftarrow x_j^{(i)}$
    $\forall j, \frac{\partial e^{(i)}}{\partial w_j} \leftarrow \frac{\partial e^{(i)}}{\partial y^{(i)}} \frac{\partial y^{(i)}}{\partial a^{(i)}} \frac{\partial a^{(i)}}{\partial w_j}$
    $\frac{\partial E}{\partial \mathbf{w}} \leftarrow \frac{\partial E}{\partial \mathbf{w}} + \frac{\partial e^{(i)}}{\partial \mathbf{w}}$
  }
  *Ajustar* $\boldsymbol{w}$*:*
  $\mathbf{w} \leftarrow \mathbf{w} - \mu \frac{\partial E}{\partial \mathbf{w}}$
} While $(E > \epsilon)$

En general la evaluación de un resultado puede pasar por distintos módulos que transforman su entrada y producen una salida. Si a partir de la entrada $\mathbf{x}$ se calcula la salida $\mathbf{y}$ como

$$y_i = f_i(x_1, x_2, \ldots, x_I)$$

entonces durante el cálculo del gradiente, la señal retropropagada es

$$\frac{\partial E}{\partial x_j} = \sum_i \frac{\partial f_i}{x_j} \frac{\partial E}{\partial y_i}.$$

Si simplificamos la notación como $\dot{\mathbf{x}} = \frac{\partial E}{\partial \mathbf{X}}$, entonces la regla de retropropagación se convierte en

$$\dot{x}_j = \sum_i \frac{\partial f_i}{x_j} \cdot \dot{y}_i.$$

Con esto podemos reescribir el algoritmo de entrenamiento de una neurona como sigue:

ENTRENAR$(X, D, \mu, \epsilon) \rightarrow \mathbf{w}$:
*Inicializar* $\boldsymbol{w}$*:*
$\mathbf{w} \leftarrow$ VECTORALEATORIO$(I + 1)$
*Iterar hasta que* $E$ *sea inferior a* $\epsilon$*:*
Do {
  *Inicializar* $E$, $\partial E/\partial \boldsymbol{w}$*:*
  $E \leftarrow 0$, $\partial E/\partial \mathbf{w} \leftarrow \mathbf{0}$
  *Calcular* $E$, $\partial E/\partial \boldsymbol{w}$*:*
  For $i = 1, \ldots, N$\{
    *Calcular* $e^{(i)}$*:*
    $a^{(i)} \leftarrow \sum_{j=0}^{I} w_j x_j^{(i)}$
    $y^{(i)} \leftarrow 1/(1 + \exp -a^{(i)})$
    $e^{(i)} \leftarrow (d^{(i)} - y^{(i)})^2$
    $E \leftarrow E + e^{(i)}$
    *Calcular* $\partial e^{(i)}/\partial \boldsymbol{w}$*:*
    $\dot{y}^{(i)} \leftarrow -2(d^{(i)} - y^{(i)})$
    $\dot{a}^{(i)} \leftarrow y^{(i)}(1 - y^{(i)})\dot{y}^{(i)}$
    $\forall j, \quad \dot{w}_j^{(i)} \leftarrow x_j^{(i)} \dot{a}$
    $\dot{\mathbf{w}} \leftarrow \dot{\mathbf{w}} + \dot{\mathbf{w}}^{(i)}$
  }
  *Ajustar* $\boldsymbol{w}$*:*
  $\mathbf{w} \leftarrow \mathbf{w} - \mu \dot{\mathbf{w}}$
} While $(E > \epsilon)$