Funciones matemáticas básicas

TensorFlow proporciona varias operaciones que puede usar para agregar funciones matemáticas básicas a su gráfico.

tf.add\_n(inputs, name=None)

Agrega todos los tensores de entrada en cuanto a los elementos.

**Args:**

* **inputs**: Una lista de Tensorobjetos, cada uno con la misma forma y tipo.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensorde la misma forma y tipo que los elementos de inputs.

**Subidas:**

* **ValueError**: Si inputsno todos tienen la misma forma y tipo o la forma no se puede inferir.

tf.abs(x, name=None)

Calcula el valor absoluto de un tensor.

Dado un tensor de números reales x, esta operación devuelve un tensor que contiene el valor absoluto de cada elemento x. Por ejemplo, si x es un elemento de entrada e y es un elemento de salida, esta operación se computa y=|X|.

Vea [tf.complex\_abs()](https://www.tensorflow.org/versions/r0.12/api_docs/python/math_ops/basic_math_functions#tf_complex_abs)para calcular el valor absoluto de un número complejo.

**Args:**

* **x**: A Tensoro SparseTensorde tipo float32, float64, int32, o int64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensoro SparseTensordel mismo tamaño y tipo que xcon los valores absolutos.

tf.negative(x, name=None)

Calcula el valor numérico negativo a nivel de elemento.

Es decir, y=-X.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.sign(x, name=None)

Devuelve una indicación de elemento del signo de un número.

y = sign(x) = -1si x < 0; 0 si x == 0; 1 si x > 0.

Para números complejos, y = sign(x) = x / |x|si x != 0, de lo contrario y = 0.

**Args:**

* **x**: A Tensoro SparseTensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensoro SparseTensor, respectivamente. Tiene el mismo tipo que x.

tf.reciprocal(x, name=None)

Calcula el recíproco de x elemento-sabio.

Es decir, y=1/X.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.square(x, name=None)

Calcula el cuadrado de x elemento-sabio.

Es decir, (y = x \* x = x ^ 2).

**Args:**

* **x**: A Tensoro SparseTensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensoro SparseTensor. Tiene el mismo tipo que x.

tf.round(x, name=None)

Redondea los valores de un tensor al entero más cercano, elemento-sabio.

Redondea la mitad a par. También conocido como redondeo bancario. Si desea redondear según el modo de redondeo actual del sistema, use tf :: cint. Por ejemplo:

# 'a' is [0.9, 2.5, 2.3, 1.5, -4.5]  
tf.round(a) ==> [ 1.0, 2.0, 2.0, 2.0, -4.0 ]

**Args:**

* **x**: A Tensorde tipo float32o float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensorde la misma forma y tipo que x.

tf.sqrt(x, name=None)

Calcula la raíz cuadrada de x elemento-sabio.

Es decir, (y = \ sqrt {x} = x ^ {1/2}).

**Args:**

* **x**: A Tensoro SparseTensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensoro SparseTensor, respectivamente. Tiene el mismo tipo que x.

tf.rsqrt(x, name=None)

Calcula el recíproco de la raíz cuadrada de x elemento-sabio.

Es decir, y=1/X.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.pow(x, y, name=None)

Calcula el poder de un valor a otro.

Dado un tensor xy un tensor y, esta operación calculaXypara los elementos correspondientes en xy y. Por ejemplo:

# tensor 'x' is [[2, 2], [3, 3]]  
# tensor 'y' is [[8, 16], [2, 3]]  
tf.pow(x, y) ==> [[256, 65536], [9, 27]]

**Args:**

* **x**: A Tensorde tipo float32, float64, int32, int64, complex64, o complex128.
* **y**: A Tensorde tipo float32, float64, int32, int64, complex64, o complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor.

tf.exp(x, name=None)

Calcula exponencial de x elemento-sabio. y=miX.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.log(x, name=None)

Calcula el logaritmo natural de x elemento-sabio.

Es decir, y=Iniciar sesiónmi⁡X.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.log1p(x, name=None)

Calcula el logaritmo natural de (1 + x) elemento-sabio.

Es decir, y=Iniciar sesiónmi⁡(1+X).

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.ceil(x, name=None)

Devuelve el entero más pequeño en cuanto a los elementos en no menos de x.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.floor(x, name=None)

Devuelve el entero más grande en forma de elemento no mayor que x.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.maximum(x, y, name=None)

Devuelve el máximo de xey (es decir, x> y? X: y) elemento-sabio.

*NOTA* : Maximumadmite la transmisión. Más sobre la transmisión [aquí](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64.
* **y**: Una Tensor. Debe tener el mismo tipo que x.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.minimum(x, y, name=None)

Devuelve el mínimo de xey (es decir, x <y? X: y) elemento-sabio.

*NOTA* : Minimumadmite la transmisión. Más sobre la transmisión [aquí](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64.
* **y**: Una Tensor. Debe tener el mismo tipo que x.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.cos(x, name=None)

Calcula cos de x elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.sin(x, name=None)

Calcula el pecado de x en cuanto a los elementos.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.lbeta(x, name='lbeta')

Computa ln(|Beta(x)|), reduciendo a lo largo de la última dimensión.

Dado unidimensional z = [z\_0,...,z\_{K-1}], definimos

Beta(z) = \prod\_j Gamma(z\_j) / Gamma(\sum\_j z\_j)

Y para n + 1dimensional xcon forma [N1, ..., Nn, K], definimos lbeta(x)[i1, ..., in] = Log(|Beta(x[i1, ..., in, :])|). En otras palabras, la última dimensión se trata como el zvector.

Tenga en cuenta que si z = [u, v], entonces Beta(z) = int\_0^1 t^{u-1} (1 - t)^{v-1} dt, que define la función beta bivariante tradicional.

**Args:**

* **x**: Un rango n + 1 Tensorcon tipo float, o double.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

El logaritmo de |Beta(x)|reducción a lo largo de la última dimensión.

**Subidas:**

* **ValueError**: Si xestá vacío con rango uno o menos.

tf.tan(x, name=None)

Calcula el bronceado de x elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.acos(x, name=None)

Calcula acos de x elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.asin(x, name=None)

Calcula un valor de x elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.atan(x, name=None)

Calcula atan de x elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.lgamma(x, name=None)

Calcula el registro del valor absoluto de Gamma(x)elemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.digamma(x, name=None)

Calcula Psi, la derivada de Lgamma (el log del valor absoluto de

Gamma(x)), elemento sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.erf(x, name=None)

Calcula la función de error de Gauss de xelemento-sabio.

**Args:**

* **x**: A Tensorde SparseTensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

A Tensoro SparseTensor, respectivamente. Tiene el mismo tipo que x.

tf.erfc(x, name=None)

Calcula la función de error complementario de xelemento-sabio.

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.squared\_difference(x, y, name=None)

Devuelve (x - y) (x - y) elemento-sabio.

*NOTA* : SquaredDifferenceadmite la transmisión. Más sobre la transmisión [aquí](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: half, float32, float64, int32, int64, complex64, complex128.
* **y**: Una Tensor. Debe tener el mismo tipo que x.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.igamma(a, x, name=None)

Calcule la función Gamma incompleta regularizada inferior Q(a, x).

La función Gamma incompleta regularizada inferior se define como:

P(a, x) = gamma(a, x) / Gamma(a) = 1 - Q(a, x)

dónde

gamma(a, x) = int\_{0}^{x} t^{a-1} exp(-t) dt

es la función Gamma incompleta inferior.

Nota, arriba Q(a, x)( Igammac) es la función Gamma completa regularizada superior.

**Args:**

* **a**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **x**: Una Tensor. Debe tener el mismo tipo que a.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que a.

tf.igammac(a, x, name=None)

Calcule la función Gamma incompleta regularizada superior Q(a, x).

La función Gamma incompleta regularizada superior se define como:

Q(a, x) = Gamma(a, x) / Gamma(a) = 1 - P(a, x)

dónde

Gamma(a, x) = int\_{x}^{\infty} t^{a-1} exp(-t) dt

es la función Gama incompleta superior.

Tenga en cuenta que arriba P(a, x)( Igamma) es la función Gamma completa regularizada inferior.

**Args:**

* **a**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **x**: Una Tensor. Debe tener el mismo tipo que a.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que a.

tf.zeta(x, q, name=None)

Calcule la función zeta Hurwitz ζ(X,q).

La función zeta de Hurwitz se define como:

\zeta(x, q) = \sum\_{n=0}^{\infty} (q + n)^{-x}

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **q**: Una Tensor. Debe tener el mismo tipo que x.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.

tf.polygamma(a, x, name=None)

Calcule la función polygamma ψ(norte)(X).

La función polygamma se define como:

\psi^{(n)} (x) = \frac{d^n}{dx^n} \psi(x)

dónde ψ(X) es la función digamma

**Args:**

* **a**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **x**: Una Tensor. Debe tener el mismo tipo que a.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que a.

tf.betainc(a, b, x, name=None)

Calcule la integral beta incompleta regularizada yoX(un,segundo).

La integral beta incompleta regularizada se define como:

I\_x(a, b) = \frac{B(x; a, b)}{B(a, b)}

dónde

B(x; a, b) = \int\_0^x t^{a-1} (1 - t)^{b-1} dt

es la función beta incompleta y segundo(un,segundo)es la función beta *completa* .

**Args:**

* **a**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **b**: Una Tensor. Debe tener el mismo tipo que a.
* **x**: Una Tensor. Debe tener el mismo tipo que a.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que a.

tf.rint(x, name=None)

Devuelve el entero sabio de elemento más cercano a x.

Si el resultado está a mitad de camino entre dos valores representables, se elige el representable par. Por ejemplo:

rint(-1.5) ==> -2.0  
rint(0.5000001) ==> 1.0  
rint([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0]) ==> [-2., -2., -0., 0., 2., 2., 2.]

**Args:**

* **x**: Una Tensor. Debe ser uno de los siguientes tipos: float32, float64.
* **name**: Un nombre para la operación (opcional).

**Devoluciones:**

Una Tensor. Tiene el mismo tipo que x.