微积分与凸优化

- 机器学习中数学基础第一课

- 1 基本概念
- 2 梯度下降
- 3 凸函数
- 4 凸优化

基本概念

- 极限

极限是微积分中的基础概念，它指的是变量在一定的变化过程中，
从总的来说逐渐稳定的这样一种变化趋势以及所趋向的值（极限值）。

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

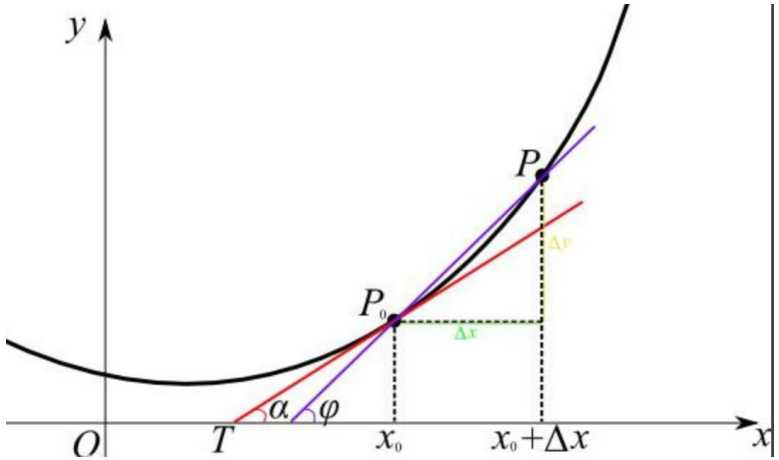
$$\lim_{x \to \infty} (1 + \frac{1}{x})^x = e$$

基本概念

- 导数

定义 设函数$y=f(x)$在点x_0的某个邻域内有定义，当自变量x在x_0处取得增量Δx（点$x_0+\Delta x$仍在该邻域内）时，相应地函数y取得增量$\Delta y=f(x_0+\Delta x)-f(x_0)$；如果$\Delta y$与$\Delta x$之比当$\Delta x\to 0$时的极限存在，则称函数$y=f(x)$在点$x_0$处**可导**，并称这个极限为函数$y=f(x)$在点$x_0$处的导数，记为$y'|_{x=x_0}$，即

- 几何意义：



基本概念

- 常用求导法则

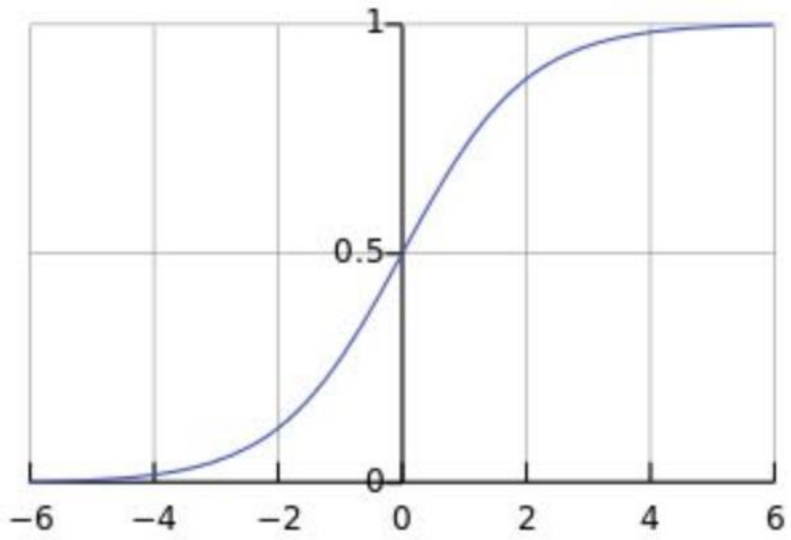
f(x)	f'(x)
ax	a
x^n	nx^{n-1}
$x+c$	1
e^x	e^x
lnx	$\frac{1}{x}$

- 求导-导数运算法则

法则	微分	偏微分
和法则(sum rule)	$(f+g)'=f'+g'$	$\frac{\partial(u+v)}{\partial x}=\frac{\partial u}{\partial x}+\frac{\partial v}{\partial x}$
积法则(product rule)	$(f\cdot g)'=f'\cdot g+f\cdot g'$	$\frac{\partial(u\cdot v)}{\partial x}=u\cdot\frac{\partial v}{\partial x}+v\cdot\frac{\partial u}{\partial x}$
链式法则(chain rule of differentiation)	$(f(g(x)))'=f'(g(x))g'(x)$	$\frac{\partial z}{\partial x}=\frac{\partial z}{\partial y}\cdot\frac{\partial y}{\partial x}$

- Sigmod函数求导：

$$函数：f(z)=\frac{1}{1+e^{-z}}$$



- 步骤

$$
\begin{aligned}
f'(z) &= \left(\frac{1}{1+e^{-z}}\right)' \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{(1+e^{-z})}\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= f(z)(1-f(z))
\end{aligned}
$$

基本概念

- 方向导数
  - 在函数定义域的内点，对某一方向求导得到的导数。

- 什么是梯度

  - 梯度的本意是一个向量（矢量），表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）。

$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

基本概念

- 1 有一个小球在曲线上滚，小球滚的方向就是梯度的方向。
  - （类比于水流从山坡流到山底）

    - y=x^2 则为斜率

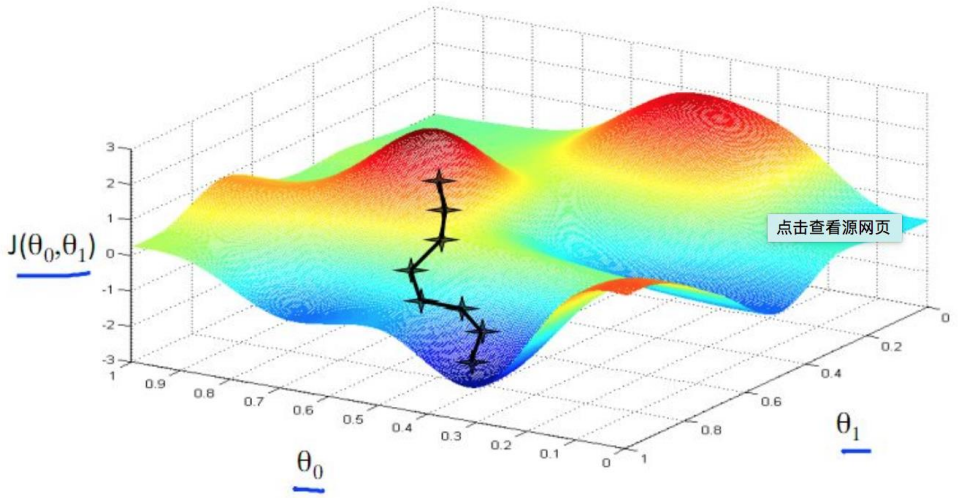
    - $f(x)=x^2 + y^2$
    - 对x偏导：2x
    - 对y偏导：2y

    - （1，1）点的话，（2，2）

基本概念

- 多维
- f(x,y,z…)=???

$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$



基本概念

- 泰勒展开：

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

  - 常用的泰勒展开式

$$e^x=1+\frac{1}{1!}x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+o(x^3)$$

$$\ln(1+x)=x-\frac{1}{2}x^2+\frac{1}{3}x^3+o(x^3)$$

$$\sin x=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5+o(x^5)$$

近似计算

梯度下降

$$\text{argmin}[(x_1+x_2-4)^2+(2x_1+3x_2-7)^2+(4x_1+x_2-9)^2]$$

考虑另外一种思维方式：迭代

$$L(x)=(x_1+x_2-4)^2+(2x_1+3x_2-7)^2+(4x_1+x_2-9)^2$$

for i in n：

1: x1=0, x2=0 $L(x)=4^2+7^2+9^2$

第二次计算的时候，我希望结果比第一次小，如何保证呢？

- 梯度下降:

迭代公式： $\theta^t = \theta^{t-1} + \Delta\theta$

将 $L(\theta^t)$ 在 θ^{t-1} 处进行一阶泰勒展开：

$$L(\theta^t) = L(\theta^{t-1} + \Delta\theta)$$
$$\approx L(\theta^{t-1}) + L'(\theta^{t-1})\Delta\theta$$

要使得 $L(\theta^t) < L(\theta^{t-1})$ ， 可取： $\Delta\theta = -\alpha L'(\theta^{t-1})$ ， 则： $\theta^t = \theta^{t-1} - \alpha L'(\theta^{t-1})$

梯度下降

- 1. 步长（Learning rate）：步长决定了在梯度下降迭代的过程中，每一步沿梯度负方向前进的长度。用上面下山的例子，步长就是在当前这一步所在位置沿着最陡峭最易下山的位置走的那一步的长度。（一般自己定义）

- 2. 特征（feature）：指的是样本中输入部分，比如样本（x_0,y_0），（x_1,y_1）,则样本特征为x，样本输出为y。

- 3. 损失函数（loss function）：为了评估模型拟合的好坏，通常用损失函数来度量拟合的程度。损失函数极小化，意味着拟合程度最好，对应的模型参数即为最优参数。在线性回归中，损失函数通常为样本输出和假设函数的差取平方。

- 4. 估计参数，希望机器学习到的，如上述的X。

梯度下降

$argmin1/2*[(x_1+x_2-4)^2+(2x_1+3x_2-7)^2+(4x_1+x_2-9)^2]$

考虑另外一种思维方式：迭代

$L(x)=(x_1+x_2-4)^2+(2x_1+3x_2-7)^2+(4x_1+x_2-9)^2$

for i in n：

1: x1=0, x2=0 $L(x)=4^2+7^2+9^2$

第二次计算的时候，我希望结果比第一次小，如何保证呢？

2: $x1=0+0.01*(-f'(x_1))$, $x_2=0+0.01*(-f'(x_2))$

$f'(x_1)=(0+0-4)*1+(2*0+3*0-7)*2+(4*0+0-9)*4=-54$

$f'(x_2)=(0+0-4)*1+(2*0+3*0-7)*3+(4*0+0-9)*1=-34$

x1=0+0.54=0.54, X2=0.34

$L_2(x) < L_1(x)$

梯度下降

- 梯度下降缺陷：
  - 这个算法有个缺点,我们算法时间复杂度,当样本较高，量比较大的话,计算量就会变得很大,所以这种方式适用的范围,仅是对那些样本较小的数据而言。

- 随机梯度下降
  - 对于每一次更新参数，不必遍历所有的训练集合，仅仅使用了一个数据，来变换一个参数。这样做不如完全梯度下降的精确度高，可能会走很多弯路，但整体趋势是走向min, 这样做可以节省更多的时间，算法更快。

梯度下降

- 随机梯度下降:

$$\text{argmin}1/2*[(x_1+x_2-4)^2+(2x_1+3x_2-7)^2+(4x_1+x_2-9)^2]$$

第二次计算的时候，我希望结果比第一次小，如何保证呢？

$$2: x1=0+0.05*(-f'(x_1)),\ x_2=0+0.05*(-f'(x_2))$$

$$f'(x_1)=(0+0-4)*1=-4$$

$$f'(x_2)=(0+0-4)*1=-4$$

$$x1=0.05*4=0.2,$$

$$x2=0.05*4=0.2$$

$$L_2(x) < L_1(x)$$

牛顿法

例1 求出函数 $f(x)=x^3-3x^2-9x+5$ 的极值.

解 $f'(x)=3x^2-6x-9=3(x+1)(x-3)$

令 $f'(x)=0$，得驻点 $x_1=-1, x_2=3$. 列表讨论

x	$(-\infty,-1)$	-1	$(-1,3)$	3	$(3,+\infty)$
$f'(x)$	+	0	−	0	+
$f(x)$	↑	极大值	↓	极小值	↑

极大值 $f(-1)=10$, 极小值 $f(3)=-22$.

将 $L(\theta^t)$ 在 θ^{t-1} 处进行二阶泰勒展开：

$$L(\theta^t)\approx L(\theta^{t-1})+L'(\theta^{t-1})\Delta\theta+L''(\theta^{t-1})\frac{\Delta\theta^2}{2}$$

为了简化分析过程，假设参数是标量（即 θ 只有一维），则可将一阶和二阶导数分别记为 g 和 h：

$$L(\theta^t)\approx L(\theta^{t-1})+g\Delta\theta+h\frac{\Delta\theta^2}{2}$$

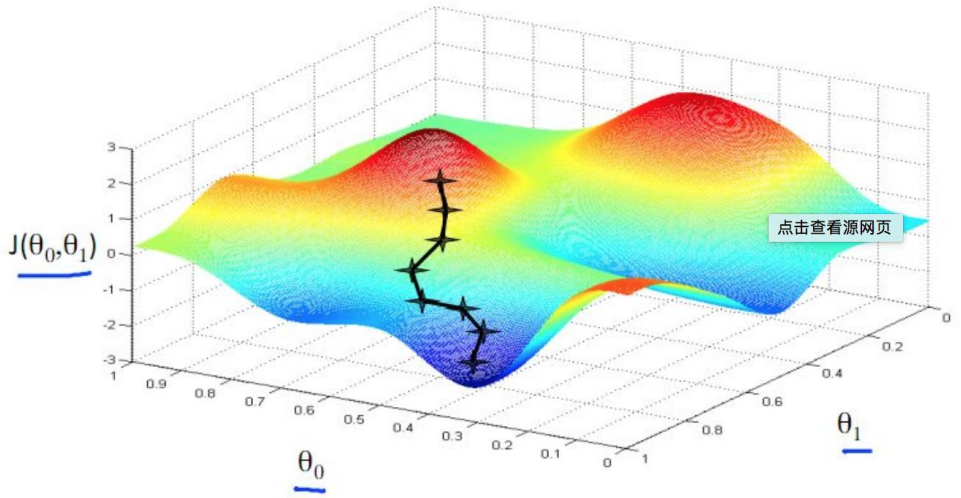
要使得 $L(\theta^t)$ 极小，即让 $g\Delta\theta+h\frac{\Delta\theta^2}{2}$ 极小，可令：$\frac{\partial\left(g\Delta\theta+h\frac{\Delta\theta^2}{2}\right)}{\partial\Delta\theta}=0$

求得 $\Delta\theta=-\frac{g}{h}$，故 $\theta^t=\theta^{t-1}+\Delta\theta=\theta^{t-1}-\frac{g}{h}$

- 最小化以下函数：x^2-2x+3

凸函数

- 一个问题:
  - 初始点不同的话，所得结果也会不同。
  - 最好的情况是:
    - 局部最小值就是全局最小值
  - 凸函数满足这个要求

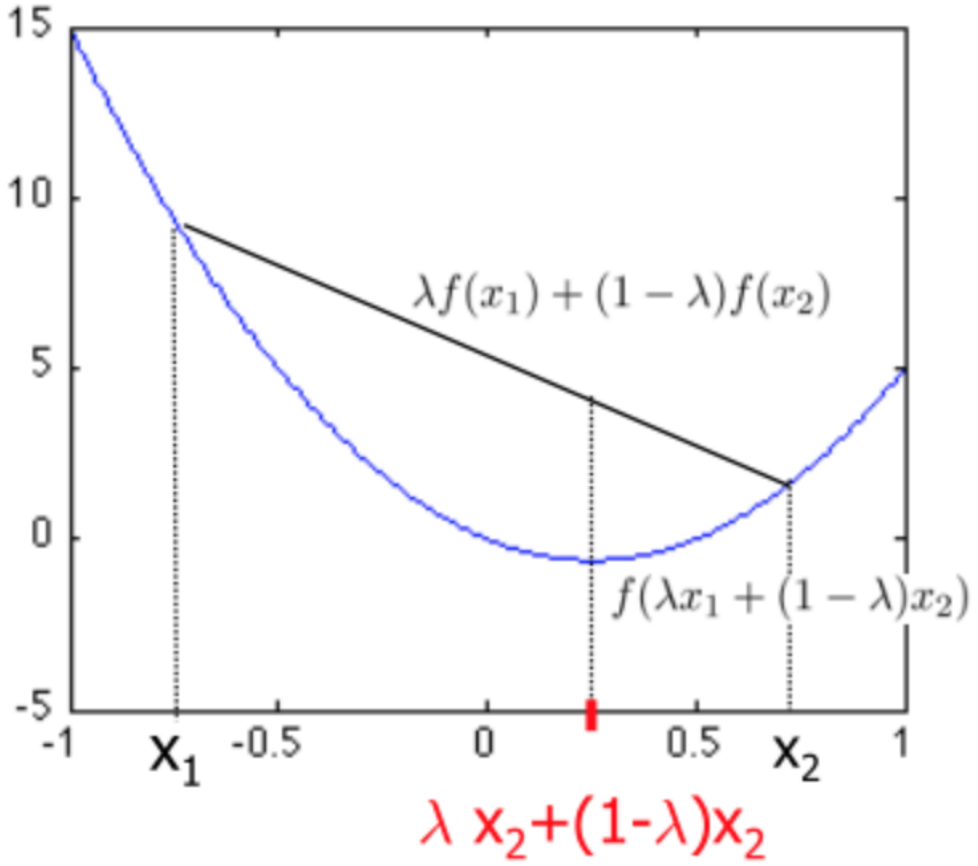


凸函数

定义：

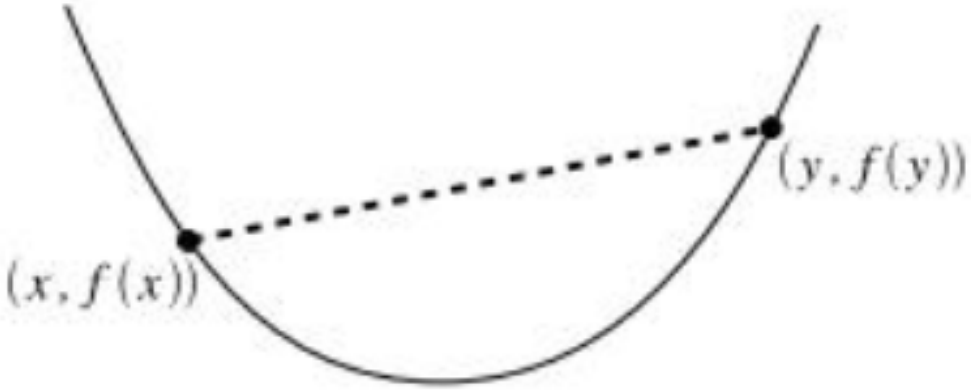
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

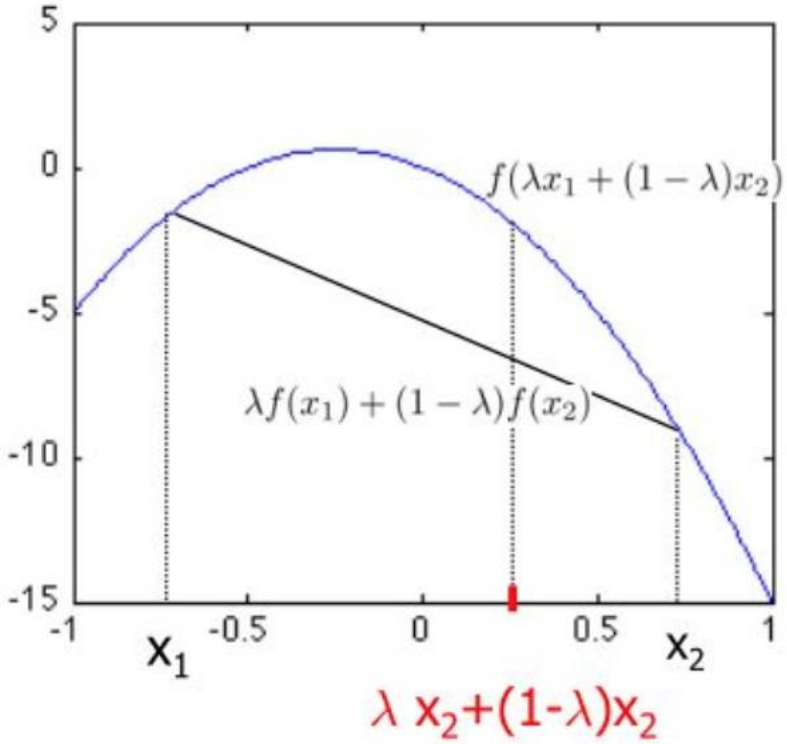
几何解释：



凸函数

那些是凸函数


$(y, f(y))$
$(x, f(x))$


5
0
-5
-10
-15
-1
-0.5
0
0.5
1
x_1
x_2
$f(\lambda x_1 + (1-\lambda)x_2)$
$\lambda f(x_1) + (1-\lambda) f(x_2)$
$\lambda x_2+(1-\lambda)x_2$

- 基本的Jensen不等式

若 $\theta_1, \ldots, \theta_k \geq 0, \theta_1 + \cdots + \theta_k = 1$

则 $f(\theta_1 x_1 + \cdots + \theta_k x_k) \leq \theta_1 f(x_1) + \cdots + \theta_k f(x_k)$

凸函数

- 保凸：
  - 非负加权求和（仿射变换）
    - f1,f2,f3 为凸， f1*w1+f2*w2+f3*w3 为凸
  - 复合函数
    - f , g 为凸， f(g(x)) 为凸
  - 逐点最大
    - f1,f2,f3为凸， max{f1,f2,f3}为凸

凸函数的例

指数函数 e^{ax}

幂函数 $x^a, x \in R_{++}, a \geq 1 \text{ or } a \leq 0.$

负对数函数 $-\log x$

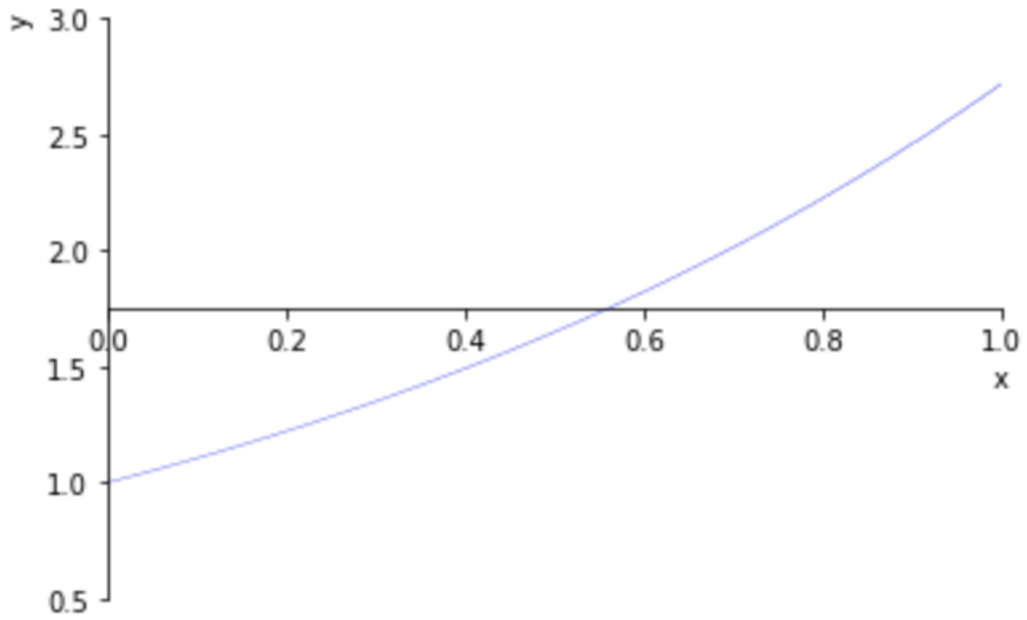
负熵函数 $x \log x$

范数函数 $\|x\|_p$

凸函数

- 指数

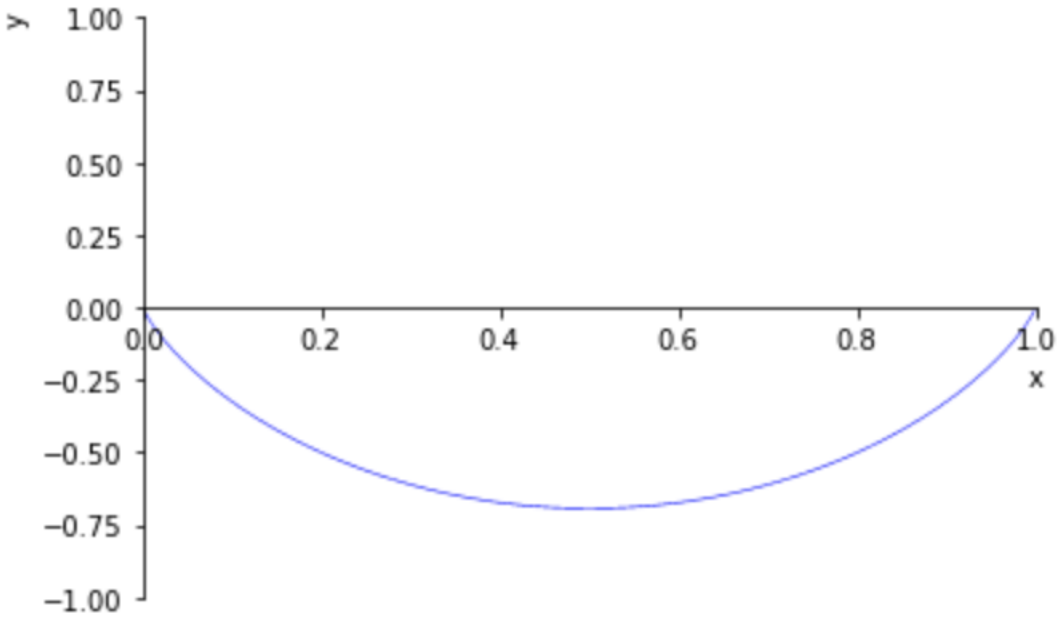
```
### exp(x)
%matplotlib inline
import numpy as np
from sympy.parsing.sympy_parser import parse_expr
from sympy import plot_implicit
from sympy import plot_implicit, cos, sin, symbols, Eq, And, log,exp
x, y = symbols('x y')
p2 = plot_implicit(Eq(exp(x), y),(x, 0, 1), (y, 0.5, 3))
```



凸函数

- 负熵

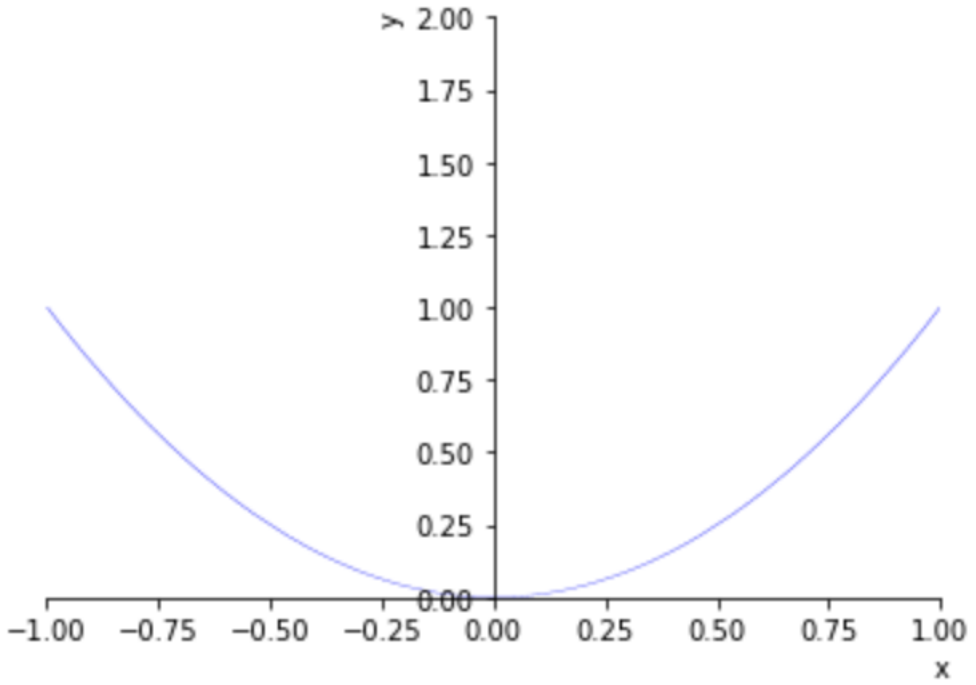
```
### x*log(x)+(1-x)*log(1-x)
%matplotlib inline
import numpy as np
from sympy.parsing.sympy_parser import parse_expr
from sympy import plot_implicit
from sympy import plot_implicit, cos, sin, symbols, Eq, And, log
x, y = symbols('x y')
p2 = plot_implicit(Eq(x*log(x)+(1-x)*log(1-x), y),(x, 0, 1), (y, -1, 1))
```



凸函数

- 范数

```
### x^2
%matplotlib inline
import numpy as np
from sympy.parsing.sympy_parser import parse_expr
from sympy import plot_implicit
from sympy import plot_implicit, cos, sin, symbols, Eq, And, log,exp,sqrt
x, y = symbols('x y')
p2 = plot_implicit(Eq(x*x, y),(x, -1, 1), (y, 0, 2))
```



凸优化

- 无约束优化问题
  - 对于x的函数f(x)，求解函数最小值：
    - $f(x,y)=x^2+y^2$

    - 1. 求偏导。
    - 2. 令偏导为0。
    - 3. 得出结果。

凸优化

- 等式约束优化问题
  - 对于x的函数f(x)，求解函数最小值，同时满足条件h(x)=0：

$$\min_{x} \ f(x) \qquad x \in R^N$$
$$s.t. \qquad h(x) = 0$$

- 例如如下问题：

$$f(x,y)=x^2y$$
$$h(x,y)=x^2+y^2-1=0$$
$$\therefore L(x,y,\lambda)=x^2y+\lambda(x^2+y^2-1)$$
$$\begin{cases}\dfrac{\partial L}{\partial x}=2xy+2\lambda x=0\\ \dfrac{\partial L}{\partial y}=x^2+2\lambda y=0\\ \dfrac{\partial L}{\partial \lambda}=x^2+y^2-1=0\end{cases}$$
$$x=0,y=\pm1,\lambda=0$$

- 更形象的例子（直线和圆相切）

一般优化问题：

$$\begin{aligned} \text{minimize} \quad & f_0(x), \quad x \in \mathbf{R}^n \\ \text{subject to} \quad & f_i(x) \le 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

拉格朗日函数：

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$