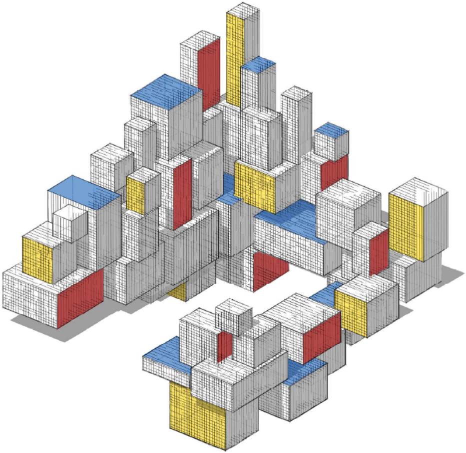
Representer Theorem



Purpose

In this lecture we discuss:

- The kernel trick
- The representer theorem
- Applications of the representer theorem
  - Surface reconstruction
  - Smoothing splines

Representer Theorem

Recall the supervised learning setting:

- We are given training data $\tau = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and a loss function that measures the fit to the data.
- We wish to find a function $g \in \mathcal{G}$ that minimizes the training loss, with the addition of a regularization term.
- We assume that the class $\mathcal{G}$ of prediction functions can be decomposed as the direct sum of an RKHS $\mathcal{H}$, defined by a kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and another linear space of real-valued functions $\mathcal{H}_0$ on $\mathcal{X}$; that is,

$$\mathcal{G} = \mathcal{H} \oplus \mathcal{H}_0,$$

meaning that any element $g \in \mathcal{G}$ can be written as $g = h + h_0$, with $h \in \mathcal{H}$ and $h_0 \in \mathcal{H}_0$. In minimizing the training loss we wish to penalize the h term of g but not the h_0 term.

Functional Optimization

The aim is to solve the functional optimization problem

$$\min_{g \in \mathcal{H} \oplus \mathcal{H}_0} \frac{1}{n} \sum_{i=1}^{n} \mathrm{Loss}(y_i, g(\boldsymbol{x}_i)) + \gamma \, \|g\|_{\mathcal{H}}^2, \tag{1}$$

where $\|g\|_{\mathcal{H}}$ means $\|h\|_{\mathcal{H}}$ if $g = h + h_0$.

In this way, we can view $\mathcal{H}_0$ as the null space of the functional $g \mapsto \|g\|_{\mathcal{H}}$.

This null space may be empty, but typically has a small dimension m; for example it could be the one-dimensional space of constant functions.

Example: Ridge Regression (cont.)

Consider again the ridge regression optimization problem

$$\min_{g \in \mathcal{H} \oplus \mathcal{H}_0} \frac{1}{n} \sum_{i=1}^{n} (y_i - g(\widetilde{\boldsymbol{x}}_i))^2 + \gamma \, \|g\|_{\mathcal{H}}^2,$$

where we have feature vectors $\widetilde{\boldsymbol{x}} = [1, \boldsymbol{x}^\top]^\top$ and $\mathcal{G}$ consists of functions of the form $g : \widetilde{\boldsymbol{x}} \mapsto \beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}$.

Each function g can be decomposed as $g = h + h_0$, where $h : \widetilde{\boldsymbol{x}} \mapsto \boldsymbol{x}^\top \boldsymbol{\beta}$, and $h_0 : \widetilde{\boldsymbol{x}} \mapsto \beta_0$.

For $g \in \mathcal{G}$, we have $\|g\|_{\mathcal{H}} = \|\boldsymbol{\beta}\|$, and so the null space $\mathcal{H}_0$ of the functional $g \mapsto \|g\|_{\mathcal{H}}$ is the set of constant functions here, which has dimension $m = 1$.

Example: Ridge Regression (cont.)

Regularization favors elements in $\mathcal{H}_0$ and penalizes large elements in $\mathcal{H}$.

As the regularization parameter γ varies between zero and infinity, solutions to (1) vary from "complex" ($g \in \mathcal{H} \oplus \mathcal{H}_0$) to "simple" ($g \in \mathcal{H}_0$).

By choosing $\mathcal{H}$ to be an RKHS in (1) this functional optimization problem effectively becomes a parametric optimization problem.

The reason is that any solution to (1) can be represented as a finite-dimensional linear combination of kernel functions, evaluated at the training sample.

This is due to the next representer theorem and is known as the kernel trick.

Theorem: Representer Theorem

Let $\mathcal{H}$ be an RKHS with kernel κ. The solution to the penalized optimization problem

$$\min_{g \in \mathcal{H} \oplus \mathcal{H}_0} \frac{1}{n} \sum_{i=1}^{n} \mathrm{Loss}(y_i, g(\boldsymbol{x}_i)) + \gamma \, \|g\|_{\mathcal{H}}^2,$$

is of the form

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{j=1}^{m} \eta_j \, q_j(\boldsymbol{x}), \tag{2}$$

where $\{q_1, \ldots, q_m\}$ is a basis of $\mathcal{H}_0$.

Proof

Let $\mathcal{F} = \text{Span}\left\{\kappa_{\boldsymbol{x}_i}, i = 1, \ldots, n\right\}$. Clearly, $\mathcal{F} \subseteq \mathcal{H}$.

Let $\mathcal{F}^{\perp}$ be the orthogonal complement of $\mathcal{F}$; that is, the class of functions

$$\{f^{\perp} \in \mathcal{H} : \langle f^{\perp}, f\rangle_{\mathcal{H}} = 0,\ f \in \mathcal{F}\} \equiv \{f^{\perp} : \langle f^{\perp}, \kappa_{\boldsymbol{x}_i}\rangle_{\mathcal{H}} = 0,\ \forall i\}.$$

Then $\mathcal{H} = \mathcal{F} \oplus \mathcal{F}^{\perp}$.

By the reproducing kernel property, for all $f^{\perp} \in \mathcal{F}^{\perp}$:

$$f^{\perp}(\boldsymbol{x}_i) = \langle f^{\perp}, \kappa_{\boldsymbol{x}_i}\rangle_{\mathcal{H}} = 0, \quad i = 1, \ldots, n.$$

Proof (cont.)

Take any $g \in \mathcal{H} \oplus \mathcal{H}_0$, and write it as $g = f + f^{\perp} + h_0$, with $f \in \mathcal{F}$, $f^{\perp} \in \mathcal{F}^{\perp}$, and $h_0 \in \mathcal{H}_0$.

By the definition of the null space $\mathcal{H}_0$, we have $\|g\|_{\mathcal{H}}^2 = \|f + f^{\perp}\|_{\mathcal{H}}^2$.

By Pythagoras, $\|f + f^{\perp}\|_{\mathcal{H}}^2 = \|f\|_{\mathcal{H}}^2 + \|f^{\perp}\|_{\mathcal{H}}^2$.

It follows that

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\mathrm{Loss}(y_i, g(\boldsymbol{x}_i)) + \gamma\|g\|_{\mathcal{H}}^2 &= \frac{1}{n}\sum_{i=1}^{n}\mathrm{Loss}(y_i, f(\boldsymbol{x}_i) + h_0(\boldsymbol{x}_i)) + \gamma\left(\|f\|_{\mathcal{H}}^2 + \|f^{\perp}\|_{\mathcal{H}}^2\right) \\ &\geqslant \frac{1}{n}\sum_{i=1}^{n}\mathrm{Loss}(y_i, f(\boldsymbol{x}_i) + h_0(\boldsymbol{x}_i)) + \gamma\,\|f\|_{\mathcal{H}}^2.\end{aligned}$$

Since we can obtain equality by taking $f^{\perp} = 0$, this implies that the minimizer of the penalized optimization problem (1) lies in the subspace $\mathcal{F} \oplus \mathcal{H}_0$ of $\mathcal{G} = \mathcal{H} \oplus \mathcal{H}_0$, and hence is of the form (2). $\square$

Parametric Optimization

Substituting the representation (2) of g into (1) gives the finite-dimensional parametric optimization problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n, \boldsymbol{\eta} \in \mathbb{R}^m} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, (\mathbf{K}\boldsymbol{\alpha} + \mathbf{Q}\boldsymbol{\eta})_i) + \gamma\, \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}, \tag{3}$$

where

- $\mathbf{K}$ is the $n \times n$ (Gram) matrix with entries $[\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j), i = 1, \ldots, n,\ j = 1, \ldots, n]$.
- $\mathbf{Q}$ is the $n \times m$ matrix with entries $[q_j(\boldsymbol{x}_i), i = 1, \ldots, n,\ j = 1, \ldots, m]$.

Convex Optimization

In particular, for the squared-error loss we have

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n,\,\boldsymbol{\eta}\in\mathbb{R}^m} \frac{1}{n}\,\|\,\boldsymbol{y}-(\mathbf{K}\boldsymbol{\alpha}+\mathbf{Q}\boldsymbol{\eta})\,\|^2+\gamma\,\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}. \tag{4}$$

This is a convex optimization problem, and its solution is found by differentiating (4) with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ and equating to zero, leading to the following system of $(n+m)$ linear equations:

$$\begin{bmatrix} \mathbf{K}\mathbf{K}^\top + n\,\gamma\mathbf{K} & \mathbf{K}\mathbf{Q} \\ \mathbf{Q}^\top\mathbf{K}^\top & \mathbf{Q}^\top\mathbf{Q} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \mathbf{K}^\top \\ \mathbf{Q}^\top \end{bmatrix} \boldsymbol{y}. \tag{5}$$

As long as $\mathbf{Q}$ is of full column rank, the minimizing function is unique.

Example: Ridge Regression (cont.)

Recall the ridge regression minimization program:

$$\min_{g \in \mathcal{H} \oplus \mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} (y_i - g(\widetilde{\boldsymbol{x}}_i))^2 + \gamma \, \|g\|_{\mathcal{H}}^2, \tag{6}$$

which we rewrote as:

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \| \boldsymbol{y} - \beta_0 \mathbf{1} - \mathbf{X}\boldsymbol{\beta} \|^2 + \gamma \, \|\boldsymbol{\beta}\|^2, \tag{7}$$

In this case, $\mathcal{H}$ is the RKHS with linear kernel function $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$ and $\mathcal{C} = \mathcal{H}_0$ is the linear space of constant functions, which is spanned by the function $q_1 \equiv 1$. Moreover, $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ and $\mathbf{Q} = \mathbf{1}$.

If we appeal to the representer theorem directly, then the problem in (6) becomes, as a result of (3):

$$\min_{\boldsymbol{\alpha}, \eta_0} \frac{1}{n} \left\| \boldsymbol{y} - \eta_0 \, \mathbf{1} - \mathbf{X}\mathbf{X}^\top \boldsymbol{\alpha} \right\|^2 + \gamma \, \|\mathbf{X}^\top \boldsymbol{\alpha}\|^2.$$

Example: Ridge Regression (cont.)

This is a convex optimization problem, and so the solution follows by taking derivatives and setting them to zero. This gives the equations

$$\mathbf{X}\mathbf{X}^\top\big((\mathbf{X}\mathbf{X}^\top + n\,\gamma\,\mathbf{I}_n)\,\boldsymbol{\alpha} + \eta_0\,\mathbf{1} - \boldsymbol{y}\big) = 0,$$

and

$$n\,\eta_0 = \mathbf{1}^\top(\boldsymbol{y} - \mathbf{X}\mathbf{X}^\top\boldsymbol{\alpha}).$$

Note that these are equivalent to the equations we found by directly minimizing (7) (assuming that $n \geqslant p$ and $\mathbf{X}$ has full rank p). Equivalently, the solution is found by solving (5):

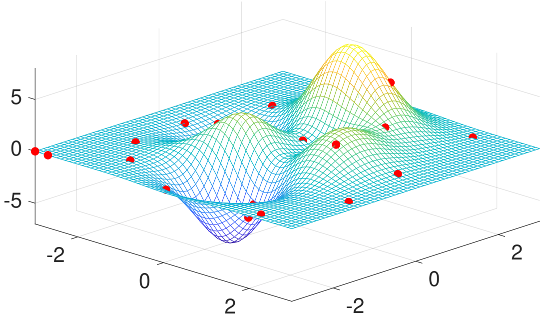
$$\begin{bmatrix} \mathbf{X}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top + n\,\gamma\,\mathbf{X}\mathbf{X}^\top & \mathbf{X}\mathbf{X}^\top\mathbf{1} \\ \mathbf{1}^\top\mathbf{X}\mathbf{X}^\top & n \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ \eta_0 \end{bmatrix} = \begin{bmatrix} \mathbf{X}\mathbf{X}^\top \\ \mathbf{1}^\top \end{bmatrix}\boldsymbol{y}.$$

This is a system of $(n+1)$ linear equations, and is typically of much larger dimension than the $(p+1)$ linear equations associated with (7).

Example: Estimating the Peaks Function

The figure shows the surface plot of the *peaks* function:

$$f(x_1, x_2) = 3(1-x_1)^2 \mathrm{e}^{-x_1^2-(x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)\mathrm{e}^{-x_1^2-x_2^2} - \frac{1}{3}\mathrm{e}^{-(x_1+1)^2-x_2^2}.$$



The goal is to learn the function $y = f(\boldsymbol{x})$ based on a small set of training data (pairs of $(\boldsymbol{x}, y)$ values) indicated by red dots.

Example: Estimating the Peaks Function

We use the Gaussian kernel on $\mathbb{R}^2$, and denote by $\mathcal{H}$ the unique RKHS corresponding to this kernel. We omit the regularization term in (1), and thus our objective is to find the solution to

$$\min_{g \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - g(\boldsymbol{x}_i))^2.$$

By the representer theorem, the optimal function is of the form

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \exp\left(-\frac{1}{2} \frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{\sigma^2}\right),$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_n]^\top$ is, by (5), the solution to the set of linear equations $\mathbf{K}\mathbf{K}^\top \boldsymbol{\alpha} = \mathbf{K}\boldsymbol{y}$.

Example: Estimating the Peaks Function

Note that we are performing regression over the class of functions $\mathcal{H}$ with an implicit feature space. Due to the representer theorem, the solution to this problem coincides with the solution to the linear regression problem for which the i-th feature (for $i = 1, \ldots, n$) is chosen to be the vector $[\kappa(\boldsymbol{x}_1, \boldsymbol{x}_i), \ldots, \kappa(\boldsymbol{x}_n, \boldsymbol{x}_i)]^\top$.

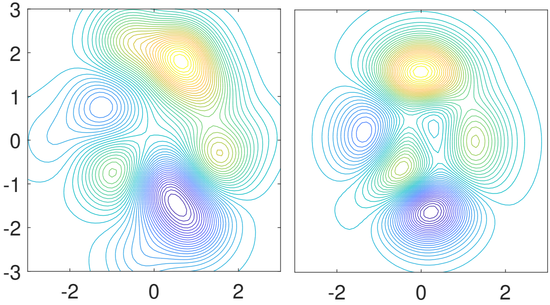


Figure: Contour plots for the prediction function g (left) and the *peaks* function (right).

peakskernel.py

```python
from genham import hammersley
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from numpy.linalg import norm

import numpy as np
def peaks(x,y):
    z =  (3*(1-x)**2 * np.exp(-(x**2) - (y+1)**2)
          - 10*(x/5 - x**3 - y**5) * np.exp(-x**2 - y**2)
          - 1/3 * np.exp(-(x+1)**2 - y**2))
    return(z)

n = 20
x = -3 + 6*hammersley([2,3],n)
z = peaks(x[:,0],x[:,1])
xx, yy = np.mgrid[-3:3:150j,-3:3:150j]
zz = peaks(xx,yy)
plt.contour(xx,yy,zz,levels=50)
fig=plt.figure()
ax = fig.add_subplot(111,projection='3d')
```

```python
ax.plot_surface(xx,yy,zz,rstride=1,cstride=1,color='c',alpha=0.3,
     linewidth=0)
ax.scatter(x[:,0],x[:,1],z,color='k',s=20)
plt.show()
sig2 = 0.3 # kernel parameter
def k(x,u):
    return(np.exp(-0.5*norm(x- u)**2/sig2))
K = np.zeros((n,n))
for i in range(n):
    for j in range(n):
        K[i,j] = k(x[i,:],x[j])
alpha = np.linalg.solve(K@K.T, K@z)

N, = xx.flatten().shape
Kx = np.zeros((n,N))
for i in range(n):
    for j in range(N):
        Kx[i,j] = k(x[i,:],np.array([xx.flatten()[j],yy.flatten()[j]]))

g = Kx.T @ alpha
dim = np.sqrt(N).astype(int)
yhat = g.reshape(dim,dim)
plt.contour(xx,yy,yhat,levels=50)
```

Smoothing Cubic Splines

In the context of data fitting, consider the optimization problem:

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \gamma \, \|g''\|^2, \tag{8}$$

where $\mathcal{G}$ is a suitable function space of twice-differentiable function from $[0, 1]$ to $\mathbb{R}$ and $\|g''\|^2 := \int_0^1 (g''(x))^2 \, \mathrm{d}x$.

In order to apply the kernel machinery, we want to write this in the form (1), for some RKHS $\mathcal{H}$ and null space $\mathcal{H}_0$.

Clearly, the norm on $\mathcal{H}$ should be of the form $\|g\|_{\mathcal{H}} = \|g''\|$ and should be well-defined.

Smoothing Cubic Splines

This suggests that we take

$$\mathcal{H} = \{g \in L^2[0,1] : \|g''\| < \infty,\ g(0) = g'(0) = 0\},$$

with inner product

$$\langle f, g \rangle_{\mathcal{H}} := \int_0^1 f''(x)\, g''(x)\, \mathrm{d}x.$$

Imposing the condition that $g(0) = g'(0) = 0$ for functions in $\mathcal{H}$ will ensure that $\mathcal{G} = \mathcal{H} \oplus \mathcal{H}_0$ where the null space $\mathcal{H}_0$ contains only affine functions, as we will see.

Smoothing Cubic Splines

To see that this $\mathcal{H}$ is in fact an RKHS, we derive its reproducing kernel. Using integration by parts, write

$$g(x) = \int_0^x g'(s)\,\mathrm{d}s = \int_0^x g''(s)\,(x-s)\,\mathrm{d}s = \int_0^1 g''(s)\,(x-s)_+\,\mathrm{d}s.$$

If κ is a kernel, then by the reproducing property it must hold that

$$g(x) = \langle g, \kappa_x \rangle_{\mathcal{H}} = \int_0^1 g''(s)\,\kappa_x''(s)\,\mathrm{d}s,$$

so that κ must satisfy $\frac{\partial^2}{\partial s^2}\kappa(x,s) = (x-s)_+$, where $y_+ := \max\{y, 0\}$. Therefore, noting that $\kappa(x,u) = \langle \kappa_x, \kappa_u \rangle_{\mathcal{H}}$, we have

$$\kappa(x,u) = \int_0^1 \frac{\partial^2 \kappa(x,s)}{\partial s^2}\frac{\partial^2 \kappa(u,s)}{\partial s^2}\,\mathrm{d}s = \frac{\max\{x,u\}\min\{x,u\}^2}{2} - \frac{\min\{x,u\}^3}{6}.$$

Smoothing Cubic Splines

This is a cubic function with quadratic and cubic terms that misses the constant and linear monomials.

If we now take $\mathcal{H}_0$ as the space of functions of the following form:

$$h_0 = \eta_1 + \eta_2\, x, \quad x \in [0, 1],$$

then (8) is exactly of the form (1).

As a consequence of the representer theorem, the optimal solution to (8) is a cubic spline with n knots:

$$g(x) = \eta_1 + \eta_2\, x + \sum_{i=1}^{n} \alpha_i\, \kappa(x_i, x). \tag{9}$$

The parameters $\boldsymbol{\alpha}, \boldsymbol{\eta}$ are determined from (3) for instance by solving (5) with matrices $\mathbf{K} = [\kappa(x_i, x_j)]_{i,j=1}^{n}$ and $\mathbf{Q}$ with i-th row of the form $[1, x_i]$ for $i = 1, \ldots, n$.

Example: Smoothing Spline

Data: $(0.05, 0.4), (0.2, 0.2), (0.5, 0.6), (0.75, 0.7), (1, 1)$.

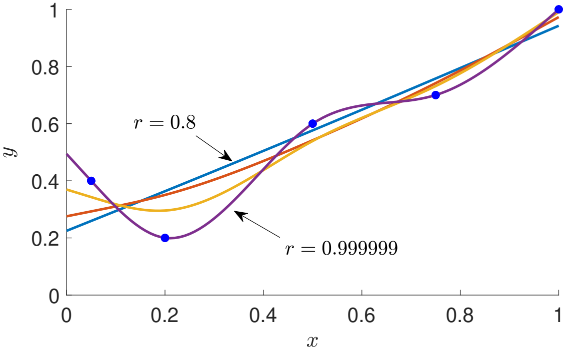


Figure: Various cubic smoothing splines for smoothing parameter $r = 1/(1 + n\,\gamma) \in \{0.8, 0.99, 0.999, 0.999999\}$. For $r = 1$, the natural spline through the data points is obtained; for $r = 0$, the simple linear regression line is found.

smoothspline.py

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([[0.05, 0.2, 0.5, 0.75, 1.]]).T
y = np.array([[0.4, 0.2, 0.6, 0.7, 1.]]).T
n = x.shape[0]
r = 0.999
ngamma = (1-r)/r

k = lambda x1, x2 : (1/2)* np.max((x1,x2)) * np.min((x1,x2)) ** 2 \
                           - ((1/6)* np.min((x1,x2))**3)
K = np.zeros((n,n))
for i in range(n):
    for j in range(n):
        K[i,j] = k(x[i], x[j])

Q = np.hstack((np.ones((n,1)), x))
m1 = np.hstack((K @ K.T + (ngamma * K), K @ Q))
m2 = np.hstack((Q.T @ K.T, Q.T @ Q))
M = np.vstack((m1,m2))

c = np.vstack((K, Q.T)) @ y

ad = np.linalg.solve(M,c)
```

```python
# plot the curve
xx = np.arange(0,1+0.01,0.01).reshape(-1,1)

g = np.zeros_like(xx)
Qx = np.hstack((np.ones_like(xx), xx))
g = np.zeros_like(xx)
N = np.shape(xx)[0]

Kx = np.zeros((n,N))
for i in range(n):
    for j in range(N):
        Kx[i,j] = k(x[i], xx[j])

g = g + np.hstack((Kx.T, Qx)) @ ad

plt.ylim((0,1.15))
plt.plot(xx, g, label = 'r = {}'.format(r), linewidth = 2)
plt.plot(x,y, 'b.', markersize=15)
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend()
```