机器学习导论-支持向量机

Introduction to Machine Learning-SVM

李文斌

https://cs.nju.edu.cn/liwenbin

liwenbin@nju.edu.cn

2024年04月02日

大纲

- 回顾
- 感知机
- 线性支持向量机
- 非线性支持向量机
- 多类支持向量机

大纲

- ❑ 回顾
- ❑ 感知机
- ❑ 线性支持向量机
- ❑ 非线性支持向量机
- ❑ 多类支持向量机

回顾

- **机器学习的目的是得到映射：$x \mapsto y$**

  - ✓ 类的先验概率： $p(y=i)$
  - ✓ 样本的先验概率： $p(\boldsymbol{x})$
  - ✓ 类条件概率（似然）： $p(\boldsymbol{x}|y=i)$
  - ✓ 后验概率： $p(y=i|\boldsymbol{x})$

回顾

❑ 从概率框架的角度

✓ 生成式模型（Generative models ）

- 估计$p(\boldsymbol{x}|y=i)$和$p(y=i)$，然后用贝叶斯定理求$p(y=i|\boldsymbol{x})$

✓ 判别式模型（Discriminative models ）

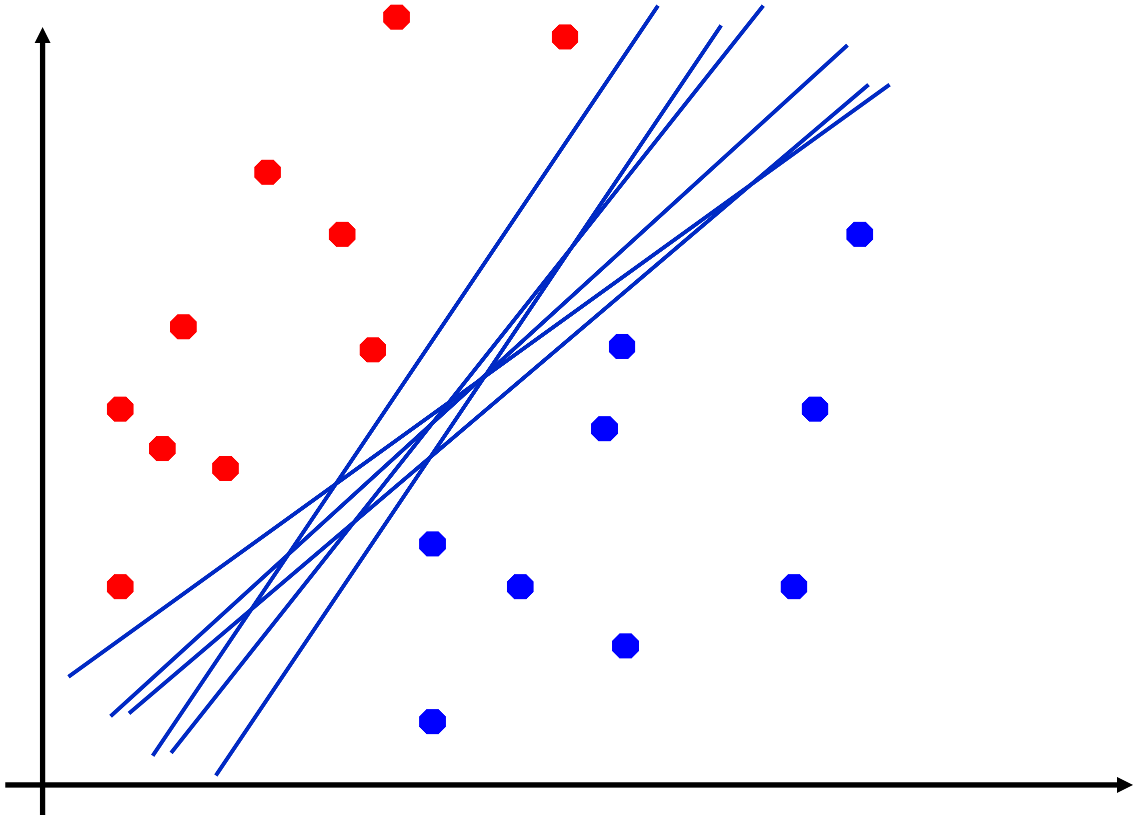
- 直接估计$p(y=i|\boldsymbol{x})$
- **判别函数**（Discriminant function）：不假设概率模型，直接求一个把各类分开的边界

大纲

- ❑ 回顾
- ❑ 感知机
- ❑ 线性支持向量机
- ❑ 非线性支持向量机
- ❑ 多类支持向量机

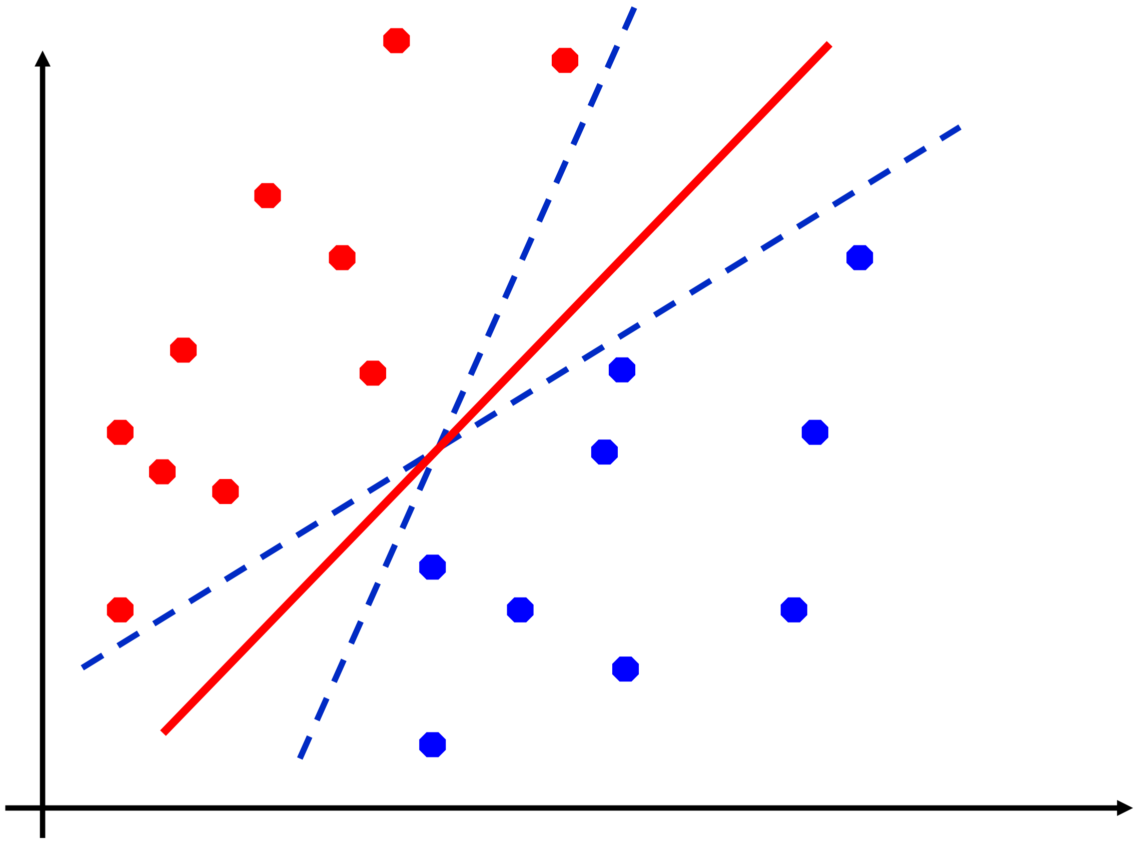
感知机

- 将训练样本分开的超平面很多，哪一个更好？



感知机

- 将训练样本分开的超平面很多，哪一个更好?



“正中间”的: 鲁棒性最好，泛化能力最强

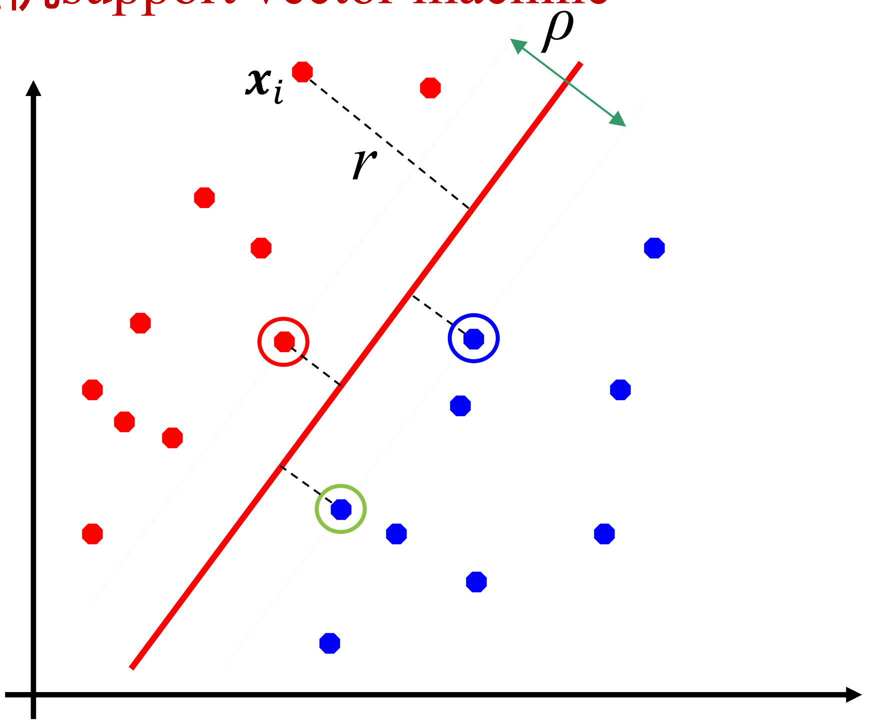
大纲

- ❑ 回顾
- ❑ 感知机
- ❑ 线性支持向量机
- ❑ 非线性支持向量机
- ❑ 多类支持向量机

线性支持向量机

❑ 间隔与支持向量

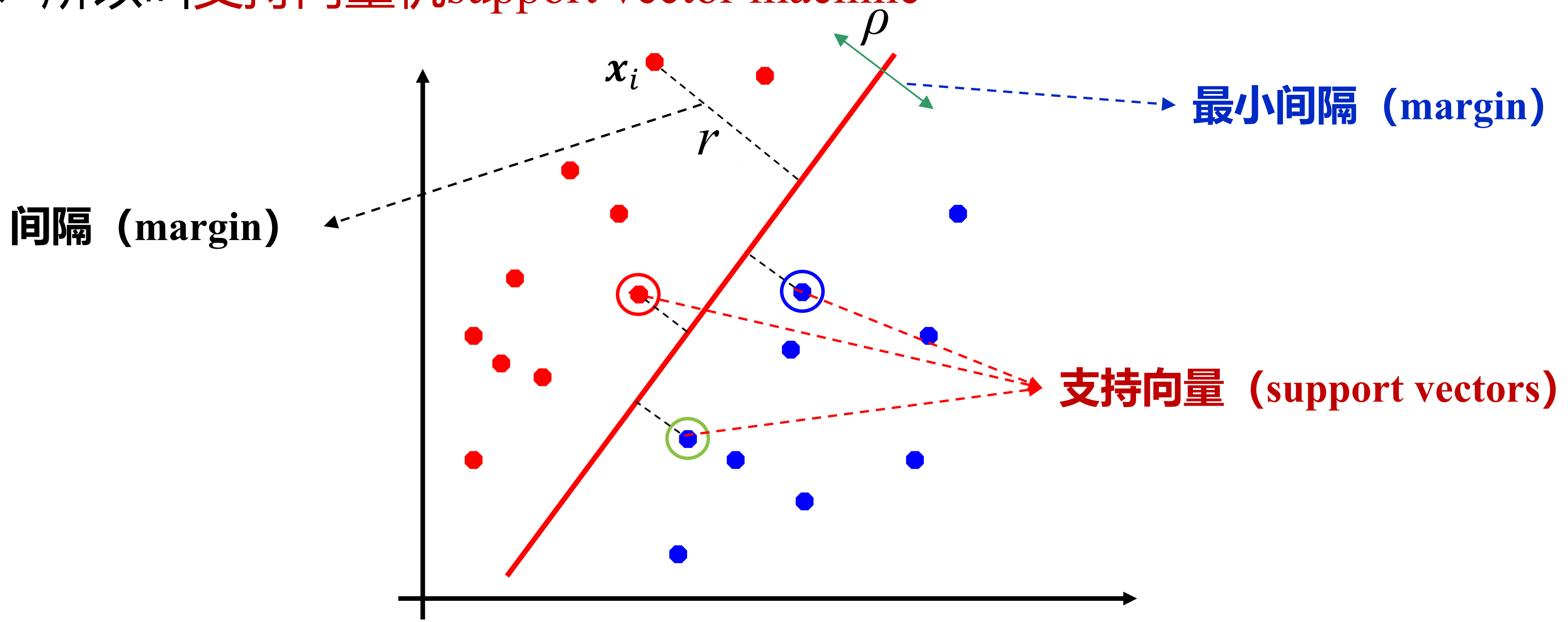
- ✓ 一个点（样例）对应的“间隔”margin是其到分界超平面的垂直距离
- ✓ SVM最大化（所有训练样本的）最小间隔margin
- ✓ 具有最小间隔的点称为支持向量(support vectors)
  - ❖ 所以叫支持向量机support vector machine



线性支持向量机

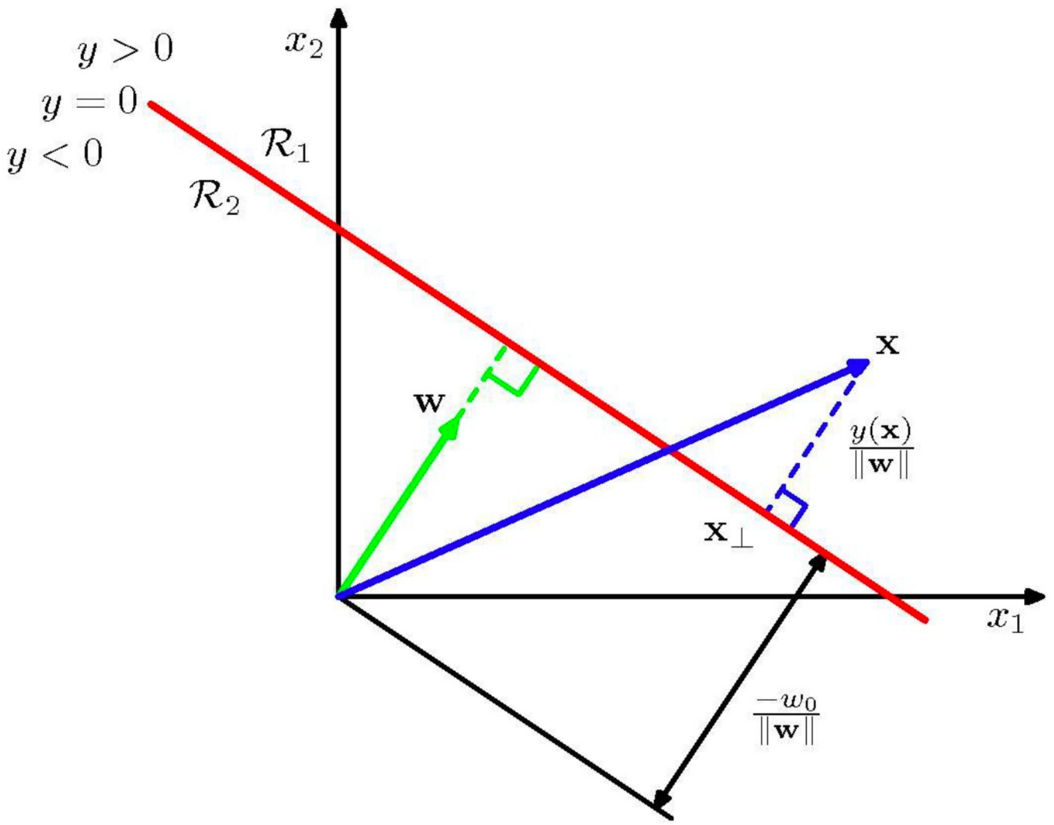
❑ 间隔与支持向量

- ✓ 一个点（样例）对应的“间隔”margin是其到分界超平面的垂直距离
- ✓ SVM最大化（所有训练样本的）最小间隔margin
- ✓ 具有最小间隔的点称为支持向量(support vectors)
  - ❖ 所以叫支持向量机support vector machine



线性支持向量机

❑ 几何示意图



- 分类超平面（红色）

  $$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$$

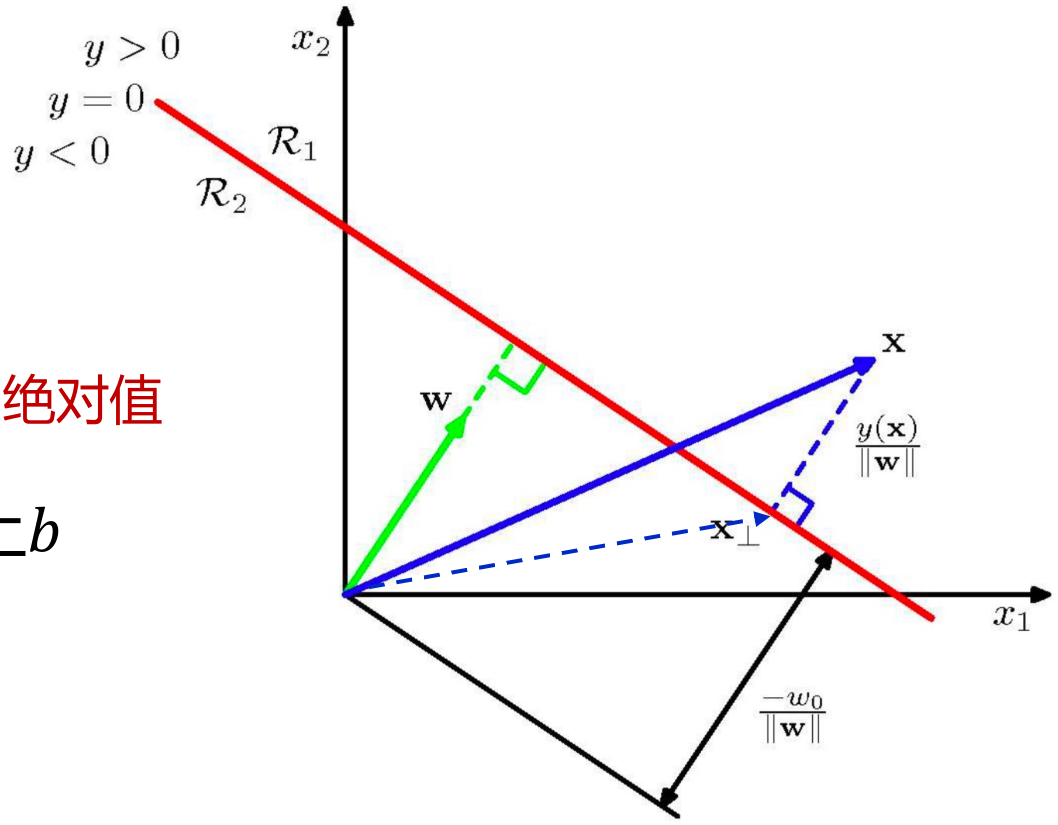
- 绿色 $\boldsymbol{w}$ 为其法向量（normal vector）

- $\boldsymbol{x}$为任一点/样例，其到超平面的距离r为?

线性支持向量机

❑ 计算Margin

- ✓ $\boldsymbol{x}$的投影点为$\boldsymbol{x}_\perp$， $\boldsymbol{x}-\boldsymbol{x}_\perp$为距离向量
  - 其方向与$\boldsymbol{w}$相同，为$\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$
  - 其大小r可为0，或正，或负; margin为其大小的绝对值
- ✓ $\boldsymbol{x}=\boldsymbol{x}_\perp+r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$，两边同乘以$\boldsymbol{w}^T$，然后加上$b$
  - $\boldsymbol{w}^T\boldsymbol{x}+b=\boldsymbol{w}^T\boldsymbol{x}_\perp+\boldsymbol{b}+r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|}$
  - $f(\boldsymbol{x})=f(\boldsymbol{x}_\perp)+r\|\boldsymbol{w}\|$ 为什么?
  - $r=\frac{f(x)}{\|\boldsymbol{w}\|}$ 为什么?



$f(\boldsymbol{x}_\perp)=0$

线性支持向量机

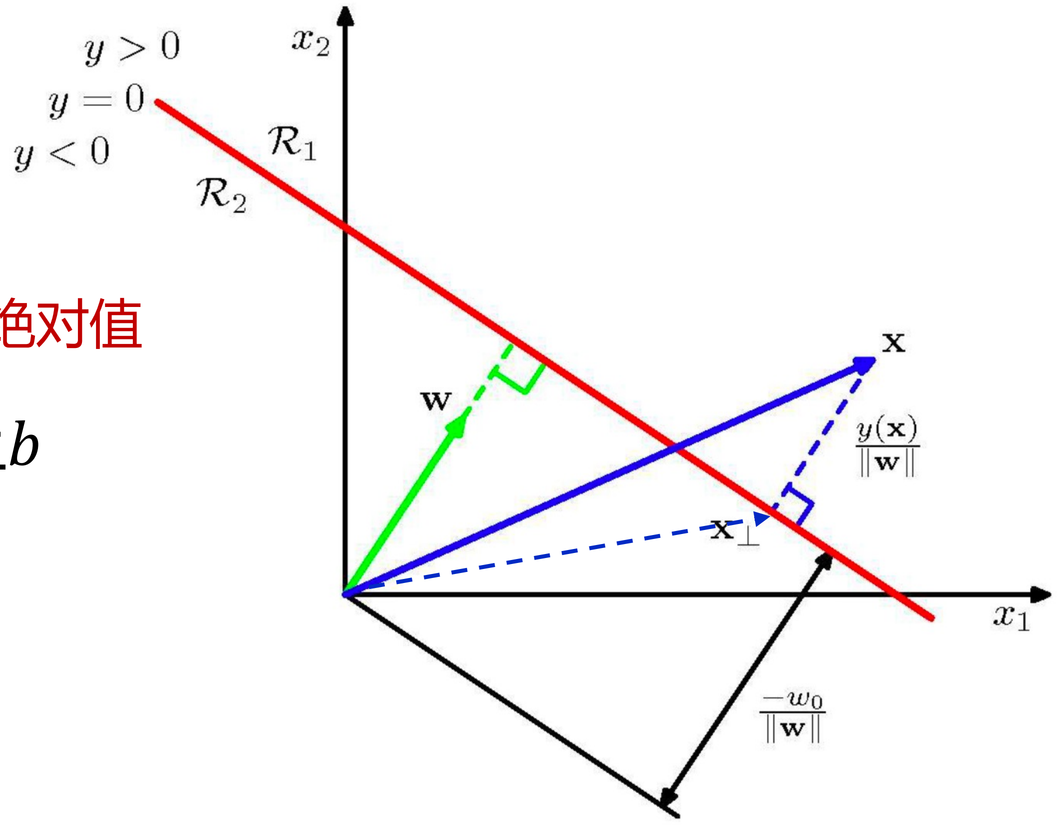
❑ 计算Margin

✓ $\boldsymbol{x}$的投影点为$\boldsymbol{x}_{\perp}$， $\boldsymbol{x}-\boldsymbol{x}_{\perp}$为距离向量

- 其方向与$\boldsymbol{w}$相同，为$\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$
- 其大小r可为0，或正，或负; margin为其大小的绝对值

✓ $\boldsymbol{x}=\boldsymbol{x}_{\perp}+r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$，两边同乘以$\boldsymbol{w}^{\boldsymbol{T}}$，然后加上$b$

- $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}+b=\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\perp}+\boldsymbol{b}+r\frac{\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}}{\|\boldsymbol{w}\|}$
- $f(\boldsymbol{x})=f(\boldsymbol{x}_{\perp})+r\|\boldsymbol{w}\|$ 为什么?
- $r=\frac{f(x)}{\|\boldsymbol{w}\|}$ 为什么?



✓ $\boldsymbol{x}$的margin是$\frac{|f(x)|}{\|\boldsymbol{w}\|}=\frac{|\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$

$f(\boldsymbol{x}_{\perp})=0$

线性支持向量机

❑ 分类与评价

- ✓ **怎样分类?**
  - ○ $f(\boldsymbol{x}) > 0$----分为正类； $f(\boldsymbol{x}) < 0$----分为负类
  - ○ 那么$f(\boldsymbol{x}) = 0$　怎么办?
- ✓ **对于任何一个样例，判断预测的对错?** （ $y_i = \{1, \ -1\}$）
  - ○ $y_i f(\boldsymbol{x}_i) > 0$----**正确**； $y_i f(\boldsymbol{x}_i) < 0$----错误
  - ○ 如果我们假设能完全分开，并且$|y_i| = 1$，那么

$$y_i f(\boldsymbol{x}_i) = |f(\boldsymbol{x}_i)|$$

线性支持向量机

❑ SVM的形式化描述

✓ SVM的问题是什么?

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\min_i\left(\frac{|\boldsymbol{w}^T\boldsymbol{x}_i+b|}{\|\boldsymbol{w}\|}\right)\right) \dashrightarrow r=\frac{|f(x)|}{\|\boldsymbol{w}\|}=\frac{|\boldsymbol{w}^T\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$$

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\min_i\left(\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|}\right)\right) \dashrightarrow y_i f(\boldsymbol{x}_i)=|f(\boldsymbol{x}_i)|$$

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\frac{1}{\|\boldsymbol{w}\|}\min_i\left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)\right)\right)$$

✓ 非常难以优化，怎么办?

- 继续简化

线性支持向量机

❑ 换个角度看问题

✓ 到目前为止

- 对w没有限制，要求最大化最小的间隔，难优化

✓ 判断对错：如果$yf(x) > 0$即正确

- 即$y(\boldsymbol{w}^T\boldsymbol{x} + b) > 0$，**只需要方向，完全不需要大小!**
- 如果$(\boldsymbol{w}, b)$变为$(c\boldsymbol{w}, cb)$，预测和间隔会变吗?

线性支持向量机

❑ 换个角度看问题

✓ 选择一个合适$c > 0$，使得$\|w\| = 1$

$$\max_{\boldsymbol{w},b} \quad \min_{1\le i\le n} y_i f(\boldsymbol{x}_i)$$

$$\text{s.t.} \quad y_i f(\boldsymbol{x}_i) > 0, \qquad 1 \le i \le n$$

$$\boldsymbol{w}^T\boldsymbol{w} = 1\,.$$

仍然难求解！

✓ 假设某个最优解为$(\boldsymbol{w}^\star, b^\star)$，选择$c$为

$$c = \min_{1\le i\le n} y_i((\boldsymbol{w}^\star)^T\boldsymbol{x} + b^\star)$$

两边同时除以c

✓ 那么$\frac{1}{c}(\boldsymbol{w}^\star, b^\star)$也是一个最优解

$$\min_{1\le i\le n} y_i\left(\left(\frac{1}{c}\boldsymbol{w}^\star\right)^T\boldsymbol{x} + \frac{1}{c}b^\star\right) = 1 > 0$$

线性支持向量机

❑ 换个角度看问题

✓ 我们限定$\min_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)$为1，不改变最优目标值

○ 问题变为：在限制$\min_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)$为1时，最大化$\frac{1}{\|\boldsymbol{w}\|}$

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\min_i\left(\frac{|\boldsymbol{w}^T\boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|}\right)\right)$$

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\min_i\left(\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}\right)\right)$$

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}}\left(\frac{1}{\|\boldsymbol{w}\|}\boxed{\min_i\left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)}\right) \longrightarrow 1$$

$$s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \forall i$$

线性支持向量机

❑ 换个角度看问题

- ✓ 我们限定$\min_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)$为1，不改变最优目标值
  - ○ 问题变为：在限制$\min_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)$为1时，最大化$\frac{1}{\|\boldsymbol{w}\|}$

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \forall i$$

线性支持向量机

❑ 拉格朗日乘子法

- ✓ $L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} a_i \left(y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) - 1\right)$
  - Subject to $a_i \geq 0$
- ✓ 证明最优化的必要条件

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{0} \quad \Rightarrow \quad \boxed{\boldsymbol{w} = \sum_{i=1}^{n} a_i\, y_i \boldsymbol{x}_i}$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad 0 = \sum_{i=1}^{n} a_i\, y_i$$

- ✓ 在此两条件下，将两个等式代入回L

$$\tilde{L}(\boldsymbol{a}) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \boxed{\boldsymbol{x}_i^T \boldsymbol{x}_j}$$ 内积

线性支持向量机

❑ Karush-Kuhn-Tucker条件，KKT

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\boxed{\alpha_i \left(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1\right) = 0 \qquad i = 1, 2, \ldots, n}$$ 互补松弛性质

$$\alpha_i \geq 0 \qquad i = 1, 2, \ldots, n$$

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 \qquad i = 1, 2, \ldots, n$$

✓ **一般而言，KTT条件不是充分必要的；但在原始SVM问题里，这些条件对于确定最优解即是充分又必要的**

线性支持向量机

❑ SVM的对偶形式

✓ 在原来的空间（即输入空间）中

- 变量是$\boldsymbol{x}_i$，称为SVM的原始形式（primal form）

✓ 现在的问题里面

- 变量是a_i，即拉格朗日乘子，称为对偶空间（dual space）
- 对偶空间完成优化后，得到最优的$\boldsymbol{a}^{\star}$，可以得到原始空间中的最优解$\boldsymbol{w}^{\star}$

✓ SVM的对偶形式（dual form）

$$\underset{\boldsymbol{a}}{\arg\max} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$s.t. \quad a_i \geq 0$$

$$\sum_{i=1}^{n} a_i y_i = 0 \longrightarrow \boxed{\frac{\partial L}{\partial b} = 0 \rightarrow 0 = \sum_{i=1}^{n} a_i y_i}$$

线性支持向量机

❑ 最优$b^\star$和支持向量

✓ 对偶形式下求解$\boldsymbol{w}^\star$

- 对偶空间完成优化后，得到最优的$\boldsymbol{a}^\star$，可以得到原始空间中的最优解$\boldsymbol{w}^\star$

$$\boldsymbol{w}^\star=\sum_{i=1}^{n} a_i^\star y_i \boldsymbol{x}_i$$

✓ 对偶形式下求解$b^\star$

互补松弛性质 $\quad a_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1)=0 \qquad i=1,2,\ldots,n$

若存在 $a_i>0$： $\quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1=0$

线性支持向量机

❑ 最优$b^\star$和支持向量

✓ 对偶形式下求解b*

互补松弛性质 $a_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1)=0 \qquad i=1,2,\ldots,n$

若存在 $a_i>0$： $y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1=0$

因此，对于特定i： $b_i^\star=y_i-(\boldsymbol{w}^\star)^T\boldsymbol{x}_i$

所有 $a_i>0$的样本，即支持向量：

$$b^\star=\frac{1}{\mathrm{n}}\sum_{a_i>0}b_i^\star$$

线性支持向量机

❑ 剩下的问题

- ✓ 如何最优化?
  - 对偶空间中
  - 原始空间中

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \forall i$$

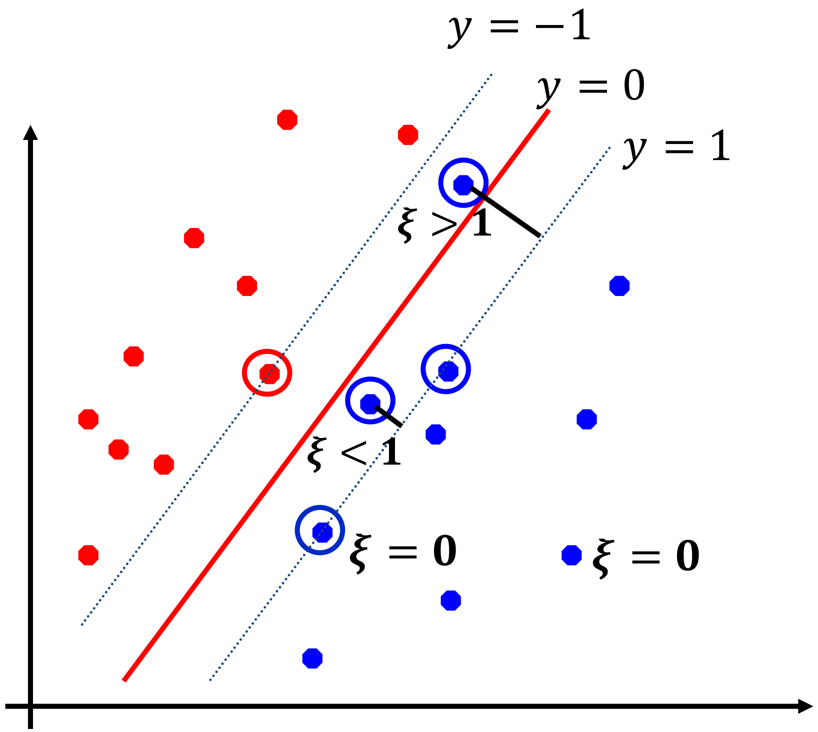
- ✓ 如果能允许少数点$y_i f(\boldsymbol{x}_i) < 1$
  - 如果允许一个点$y_i f(\boldsymbol{x}_i) < 1$，但是大幅度增加margin呢?
- ✓ 如果不是线性可分的（linearly separable），但是可以用非线性的（non-linearly separable）边界分开?
- ✓ 如果不是两个类，而是多个呢?

线性支持向量机

❑ Soft margin

✓ 可以允许少数点margin比1小

- 但是犯错误是有惩罚的，否则?
- $y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 \rightarrow y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i$
- ξ_i: 松弛变量（slack variable），即允许犯的错误
- $\xi_i \geq 0$
- 右图$\xi = 0$，> 1各自代表什么?



线性支持向量机

❑ 如何惩罚?

- ✓ 原始空间（Primal space）

$$\begin{aligned} \underset{\boldsymbol{w},b,\xi}{\operatorname{argmin}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

- ✓ $C > 0$: 正则化参数（regularization parameter）
  - ξ_i ---- 代价，我们要最小化代价函数（总代价）
  - $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$ ---- 正则项（ regularization term），对分类器进行限制，使得模型复杂度不至于太高（另一个角度，还是最大化间隔）
  - 那么，如何确定C的值?

线性支持向量机

❑ Soft margin的对偶形式

$$\underset{\boldsymbol{a}}{\operatorname{argmax}} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$s.t. \qquad C \geq a_i \geq 0$$

$$\sum_{i=1}^{n} a_i\, y_i = 0$$

✓ 对偶形式仅依赖于内积!

线性支持向量机

❑ 总结

✓ 分类器是一个分离超平面（separating hyperplane ）

✓ 最重要的训练样本是“支持向量”，它们定义了超平面，而其他训练样本被忽略了。二次优化算法可以识别哪些样本是具有非零朗格朗日乘子a_i的支持向量

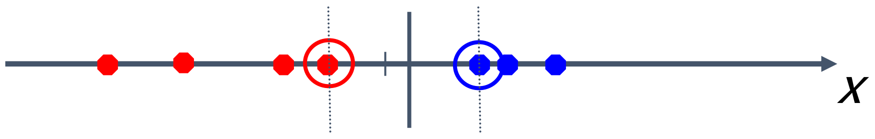
✓ 对偶问题中，训练样本只以内积形式出现。

大纲

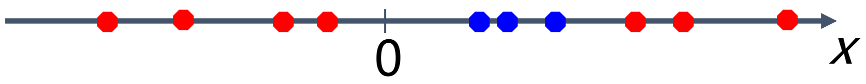
- ❑ 回顾
- ❑ 感知机
- ❑ 线性支持向量机
- ❑ 非线性支持向量机
- ❑ 多类支持向量机

非线性支持向量机

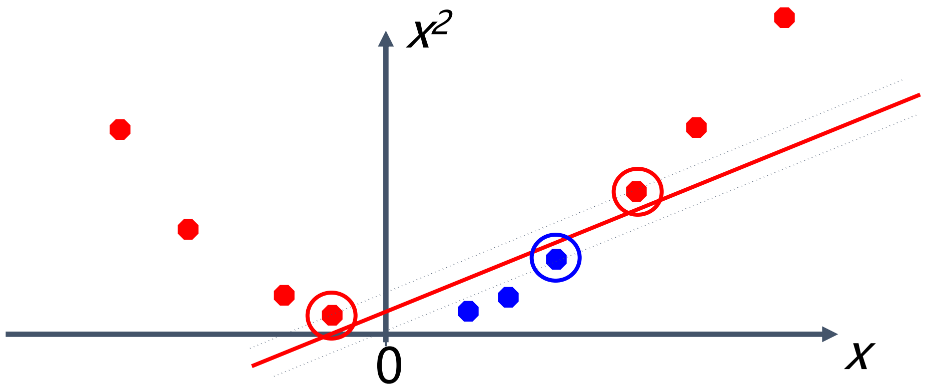
✓ 数据集是线性可分，效果会很好



✓ 数据集太困难了，怎么办?



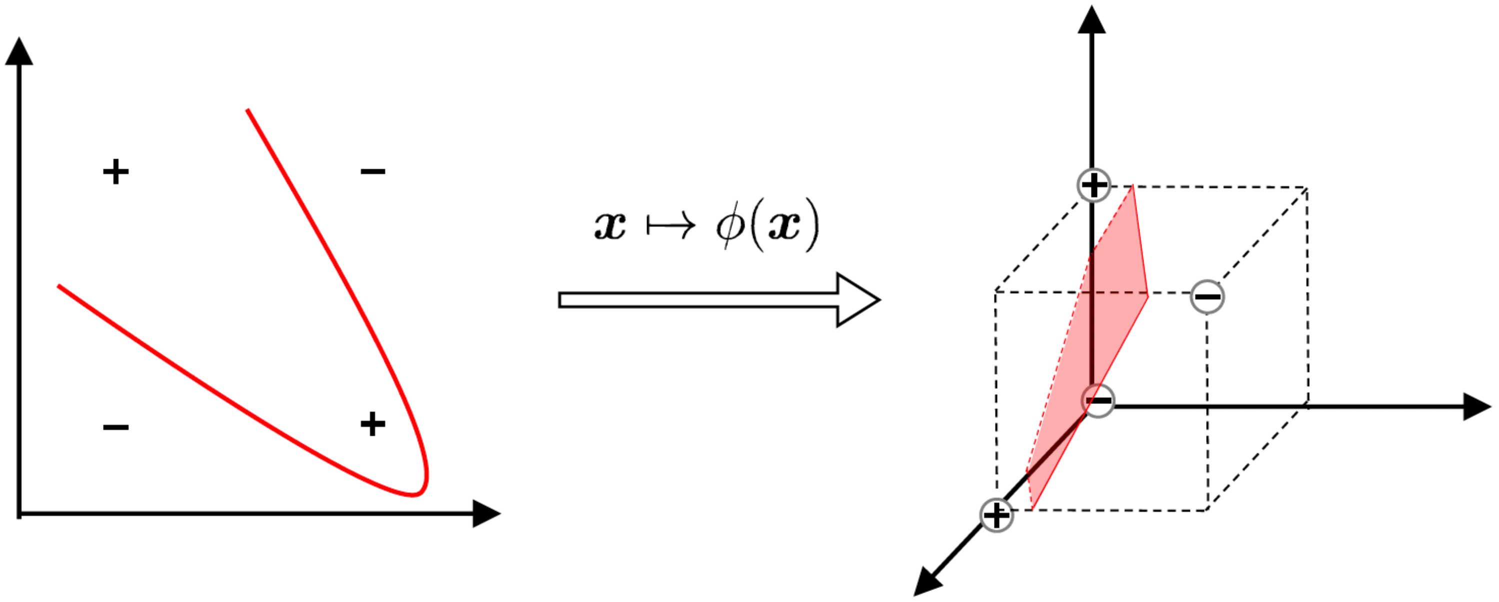
✓ 如果把数据映射到高维空间里?



非线性支持向量机

❑ 特征空间映射

✓ 将样本从原始空间映射到更高维的特征空间, 使样本在这个特征空间内线性可分



✓ 如果原始空间是有限维(属性数有限)，那么一定存在一个高维特征空间使样本可分

非线性支持向量机

❑ 内积：线性和非线性的联系

- ✓ 线性和非线性有时候紧密联系在一起----通过内积
- ✓ $\boldsymbol{x} = (x_1, x_2), \boldsymbol{z} = (z_1, z_2)$

$$K(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x}^T \boldsymbol{z})^2 = (1 + x_1 z_1 + x_2 z_2)^2$$
$$= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 x_2 z_2$$
$$= \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}^T \begin{pmatrix} 1 \\ \sqrt{2}z_1 \\ \sqrt{2}z_2 \\ z_1^2 \\ z_2^2 \\ \sqrt{2}z_1 z_2 \end{pmatrix}$$

非线性支持向量机

❑ Kernel trick

- ✓ 两个向量$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$，一个非线性函数$K(\boldsymbol{x}, \boldsymbol{y})$
- ✓ 对于满足某些条件的函数K，一定存在一个映射（mapping）$\emptyset: \mathbb{R}^d \to \Phi$，使得对任意$\boldsymbol{x}, \boldsymbol{y}$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \emptyset(\boldsymbol{x})^T \emptyset(\boldsymbol{y})$$

  - ○ 非线性函数K表示两个向量的相似程度
  - ○ 其等价于$\emptyset$里面的内积
- ✓ Φ：特征空间（feature space）
  - ○ 可以是有限维的空间，但也可以是无穷维的空间（infinite dimensional Hilbert space）

非线性支持向量机

❑ 什么样的限制条件

- ✓ 必须存在特征映射（feature mapping），才可以将非线性函数表示为特征空间中的内积
- ✓ Mercer's condition（Mercer条件，是充分必要的）：
  - ○ 对任何满足$\int g^2(\boldsymbol{u})d\boldsymbol{u} < \infty$的非零函数（平方可积函数），对称函数$K$满足条件:

$$\iint K(\boldsymbol{u},\boldsymbol{v})g(\boldsymbol{u})g(\boldsymbol{v})\,d\boldsymbol{u}d\boldsymbol{v} \geq 0$$

- ✓ 另一种等价形式:
  - ○ 对任何一个样本集合$\{x_1,\ldots,x_n\}, x_i \in \mathbb{R}^d$，如果矩阵$K = [K_{ij}]_{ij}$（矩阵的第$i$行、第$j$列元素$K_{ij} = K(x_i,x_j)$）总是半正定的，那么函数$K$满足Mercer条件

非线性支持向量机

❑ 核支持向量机Kernel SVM

- ✓ 核函数（kernel function）：K
- ✓ 对偶形式:

$$\underset{\boldsymbol{a}}{\operatorname{argmax}} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- ✓ 分类边界： $\boldsymbol{w} = \sum_{i=1}^{n} a_i\, y_i \phi(\boldsymbol{x}_i)$

- ✓ 怎样预测:

$$\boldsymbol{w}^T\phi(\boldsymbol{x}) = \phi(\boldsymbol{x})^T\left(\sum_{i=1}^{n} a_i\, y_i \phi(\boldsymbol{x}_i)\right) = \sum_{i=1}^{n} a_i\, y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

  - 线性：$\boldsymbol{w} = \sum_{i=1}^{n} a_i y_i \boldsymbol{x}_i$，$\boldsymbol{w}^T x$的计算量为$O(d)$
  - 非线性（核）方法测试所需时间为?
  - 假设计算K的时间为$O(d)$，是$O(nd)$吗?

非线性支持向量机

❑ 非线性核

- ✓ 线性核（linear kernel），dot-product kernel:

$$K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y}$$

- ✓ 非线性核（non-linear kernel）
  - ○ RBF (radial basis function)/高斯（Gaussian）核

$$K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2)$$

  - ○ 多项式核

$$K(\boldsymbol{x},\boldsymbol{y}) = (\gamma\boldsymbol{x}^T\boldsymbol{y}+c)^d$$

  - ○ 。。。

非线性支持向量机

❑ 非线性核

- ✓ $\boldsymbol{x} = (x_1, x_2), \boldsymbol{z} = (z_1, z_2)$

$\gamma = 1, c = 1, d = 2$
多项式kernel的一个特例

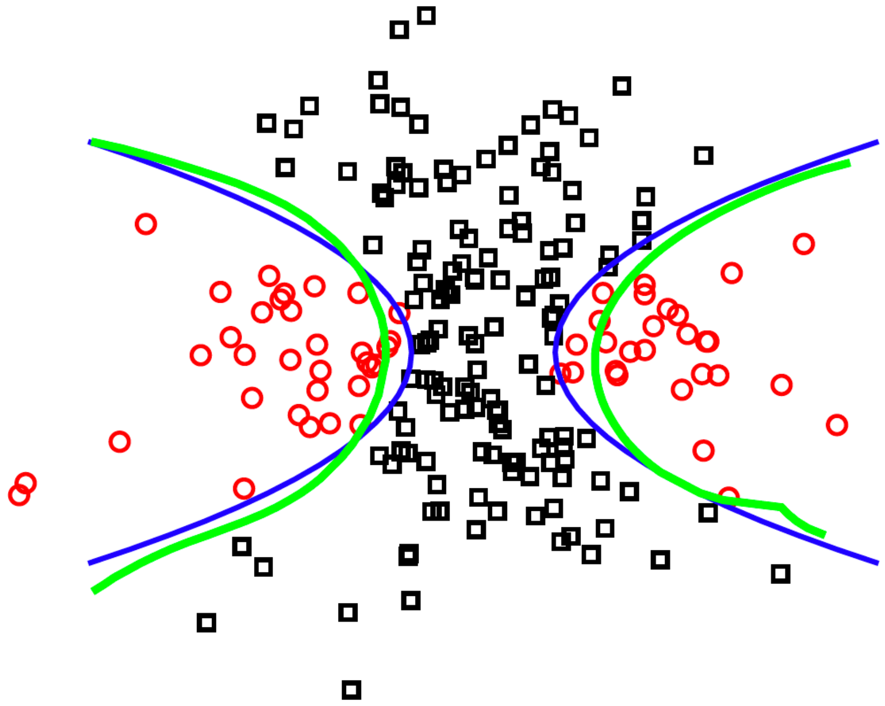
$$K(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x}^T \boldsymbol{z})^2 = (1 + x_1 z_1 + x_2 z_2)^2$$
$$= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 x_2 z_2$$
$$= \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}^T \begin{pmatrix} 1 \\ \sqrt{2}z_1 \\ \sqrt{2}z_2 \\ z_1^2 \\ z_2^2 \\ \sqrt{2}z_1 z_2 \end{pmatrix}$$

2维空间被映射到6维空间！

非线性支持向量机

❑ 非线性核的例子（RBF）

- Ground Truth boundary 真实分类边界
- 使用RBF核的SVM得到的分类边界



非线性支持向量机

❑ 超参数

- ✓ 如何决定C、γ、…
  - ○ 必须给定这些参数的值，才能进行SVM学习，SVM本身不能学习这些参数
  - ○ 称为超参数（hyper-parameter）
  - ○ 对SVM的结果有极大的影响

- ✓ 用交叉验证在训练集上学习
  - ○ 在训练集上得到不同参数的交叉验证准确率
  - ○ 选择准确率最高的超参数的数值

大纲

- ❑ 回顾
- ❑ 感知机
- ❑ 线性支持向量机
- ❑ 非线性支持向量机
- ❑ 多类支持向量机

多类支持向量机

❑ 多类（Multiclass）

- ✓ 思路：转化为2类问题
- ✓ $1 - vs. -1$（one versus one）：
  - ○ C个类$\{1,2,\ldots,C\}$
  - ○ 设计$\binom{C}{2}$个分类器，用i和j两类$(i > j)$的训练数据进行学习
  - ○ 一共$C(C-1)/2$个，其中每个类出现$C-1$次
  - ○ 对测试样本$\boldsymbol{x}$，一共会得到$C(C-1)/2$个结果，然后投票
  - ○ 对每个分类器f_i采用其二值输出，即$\mathrm{sign}(f_i(\boldsymbol{x}))$

	1	2	3
1			
2	1		
3	1	3	

多类支持向量机

❑ 多类（Multiclass）

✓ $1-vs.-all$（或$1-vs.-\mathrm{rest}$）：

- 设计C个分类器，第i个分类器用类i做正类，把其他所有$C-1$个类别的数据合并在一起做负类
  - ❖ 和交叉验证的步骤有些类似
  - ❖ 每个新的分类器f_i采用其实值输出，即$f_i(\boldsymbol{x})$
- $f_i(\boldsymbol{x})$的实数输出可以看成是属于第i类的“信心”(confidence)
- 最终选择信心最高的那个类为输出

$$\underset{i}{\operatorname{argmax}}\, f_i(\boldsymbol{x})$$

多类支持向量机

❑ 多类（Multiclass）

- ✓ 直接解决多类问题
  - Crammer-Singer方法
  - http://jmlr.org/papers/v2/crammer01a.html
- ✓ DAGSVM
  - https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/dagsvm.pdf
- ✓ ECOC
  - https://arxiv.org/pdf/cs/9501101.pdf

支持向量机的实现

✓ SVM Website: http://www.kernel-machines.org/

✓ 代表性实现

- **LIBSVM**: 非常有效且出名的SVM实现，实现了multi-class classifications, nu-SVM, one-class SVM等，而且有很多java、python等接口
- **SVM-light**: 简单但是性能不如LIBSVM，仅支持二分类，并且只用C语言实现
- **SVM-torch**: C语言实现

支持向量机的启发

❑ 从SVM的介绍学到的思想?

- ✓ 确定问题，对问题有充分的认识（实践、理论）
- ✓ 好的思路、想法idea（如margin）
  - 从理论（概率、统计？）中来
  - 从实践（已有线性分类器的缺点，如感知机）中来
- ✓ 形式化
  - 用精确的数学形式表达出来
  - 如果不能精确描述，或说明你的idea有问题
  - 开始时避免复杂模糊的想法：限制条件（如，线性可分），从较小范围开始（如，2类）
- ✓ 数学基础和研究
  - 用到的几何、凸优化、拉格朗日乘子法、Hilbert空间。。。
  - 经典的相关数学背景要熟悉：至少知道到哪里查

支持向量机的启发

❑ 简化：一种可靠的思路

- ✓ 问题（特别是数学问题）难以解决时，尽量简化
  - 问题的表述，如果难以形式化，可以将问题简化
  - 简化后的问题可以去除很多复杂的考虑，但是原问题的核心要保持
  - 如SVM从二类、线性、可分的情况开始

- ✓ 有时可以通过换思路的方法等价简化
  - 如SVM限定$\min_i \left(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)$为1
  - 也可以对原问题做不重复的修改以使简化成为可能

支持向量机的延伸

❑ 深度学习时代的SVM?

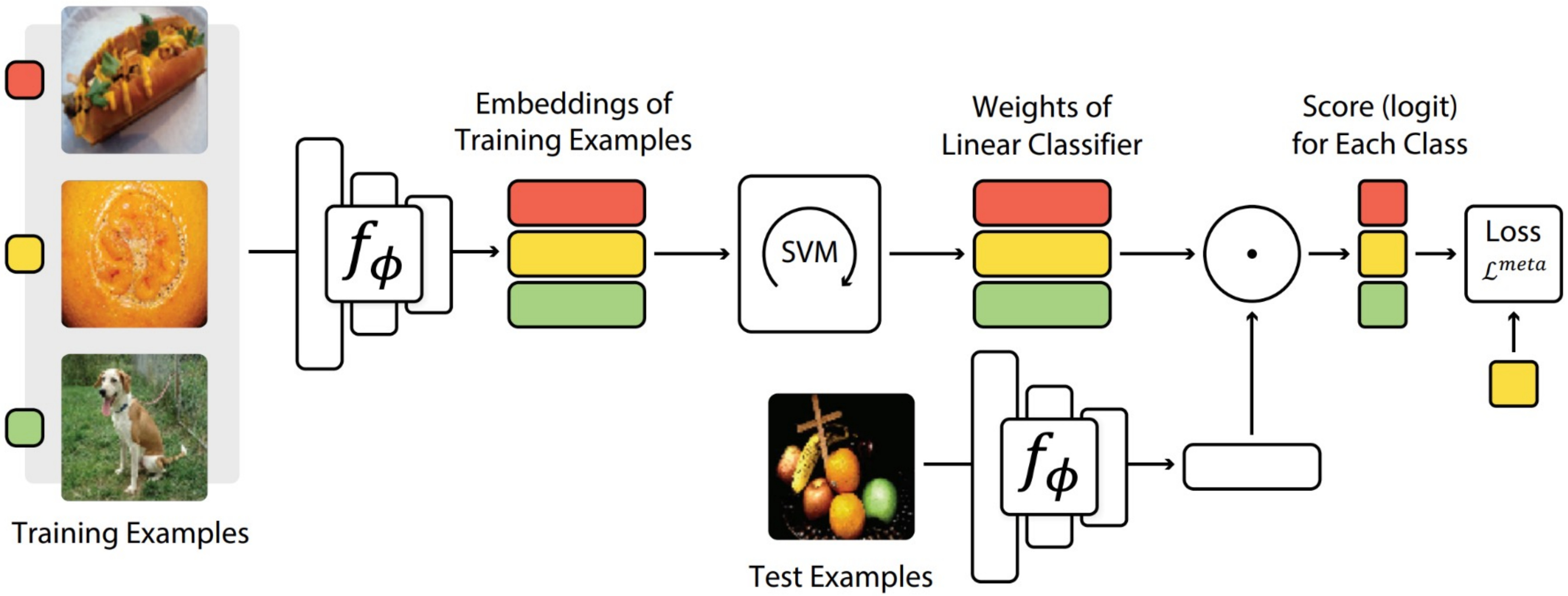


Figure 1. **Overview of our approach.** Schematic illustration of our method MetaOptNet on an 1-shot 3-way classification task. The meta-training objective is to learn the parameters ϕ of a feature embedding model f_ϕ that generalizes well across tasks when used with regularized linear classifiers (*e.g.*, SVMs). A task is a tuple of a few-shot training set and a test set (see Section 3 for details).

Lee K, Maji S, Ravichandran A, et al. Meta-learning with differentiable convex optimization. CVPR 2019.

谢 谢！

联系方式：liwenbin@nju.edu.cn

更多信息：https://cs.nju.edu.cn/liwenbin

Github介绍与使用技巧

Introduction and Usage Tips for GitHub

助教：陈怿飏

522023330016@smail.nju.edu.cn

2024年4月2日

目录

- ❑ GitHub简介
- ❑ Git基本操作
- ❑ 协作和贡献代码

- ❑ **GitHub简介**
- ❑ **Git基本操作**
- ❑ **协作和贡献代码**

GitHub简介

□ GitHub：全球最大的代码托管与协作平台

GitHub

✓ 代码托管

- GitHub允许开发人员将其项目存储在云端

✓ 版本控制

- GitHub使用Git进行版本控制，使开发人员能够跟踪项目的变化历史并管理不同版本的代码

✓ 协作开发

- 允许多名开发者同时工作在同一个项目上，支持代码审查，讨论和合并改动

✓ 社交互动

- 开发人员可以关注其他开发者、Star感兴趣的项目、参与讨论等

GitHub简介

GitHub

Git

一个**在线服务平台：**

- 使用Git作为版本控制工具来托管和协作代码
- 让使用Git进行版本控制的代码上传到云端

一个分布式**版本控制系统：**

- 用于跟踪和管理文件的更改
- 在本地机器上管理源代码历史，允许用户切换到不同的版本，合并代码改动

GitHub简介

为何需要版本控制系统?

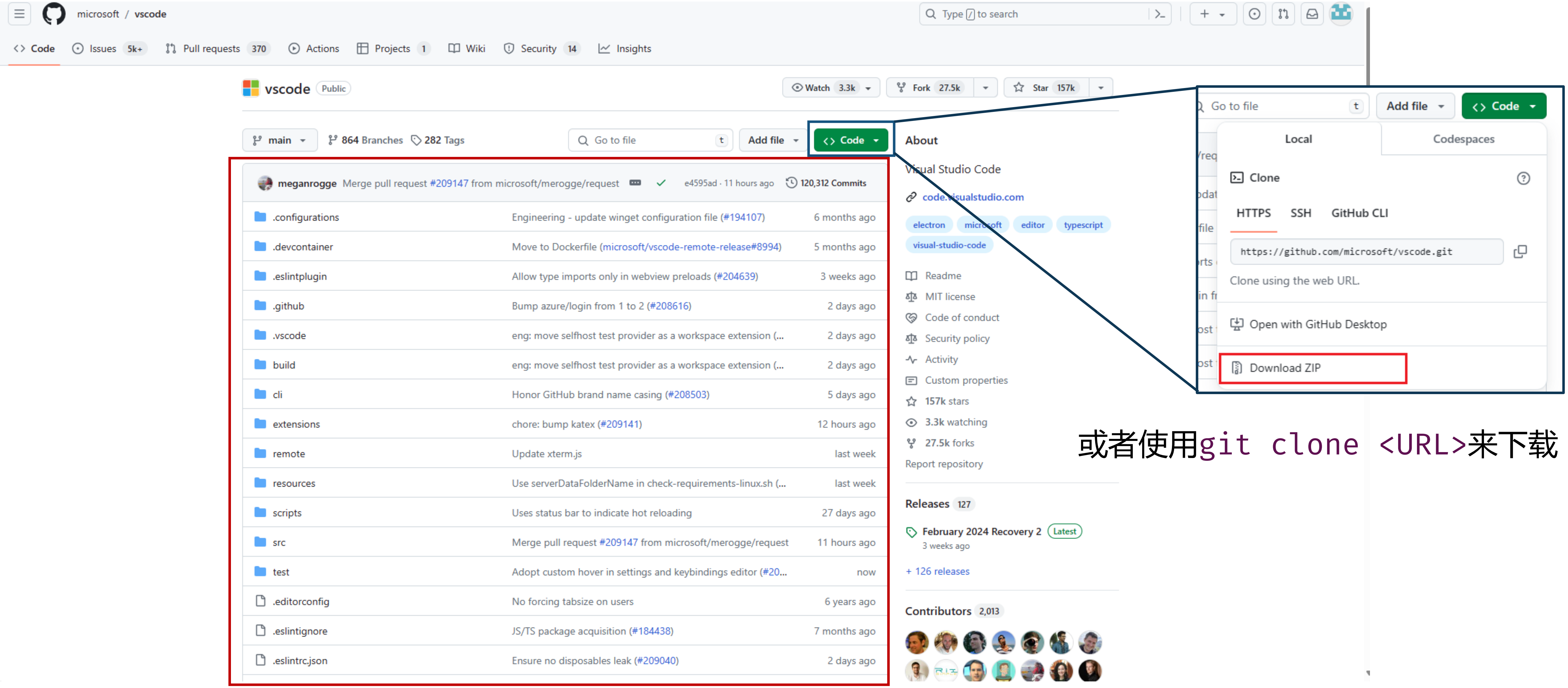
□ 并行开发:

- 团队成员可以同时工作在不同的功能上，而不会相互干扰。版本控制系统协助合并这些工作，解决代码合并时可能出现的冲突。

□ 回退错误:

- 如果引入了导致问题的代码，可以快速回到没有错误的版本。

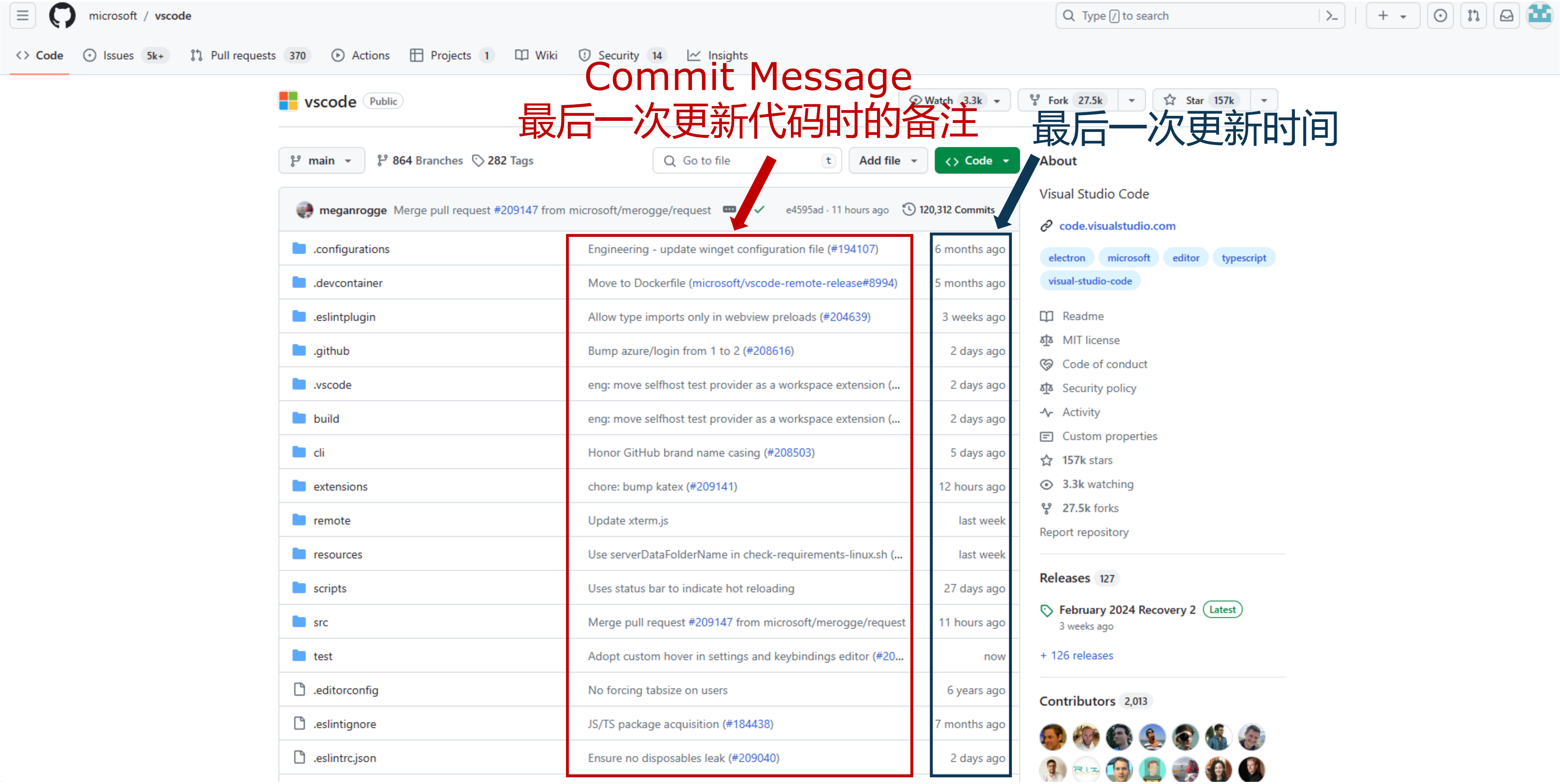
GitHub简介



或者使用`git clone <URL>`来下载

[microsoft/vscode: Visual Studio Code (github.com)]

GitHub简介



[microsoft/vscode: Visual Studio Code (github.com)]

GitHub简介

README | Code of conduct | MIT license | Security

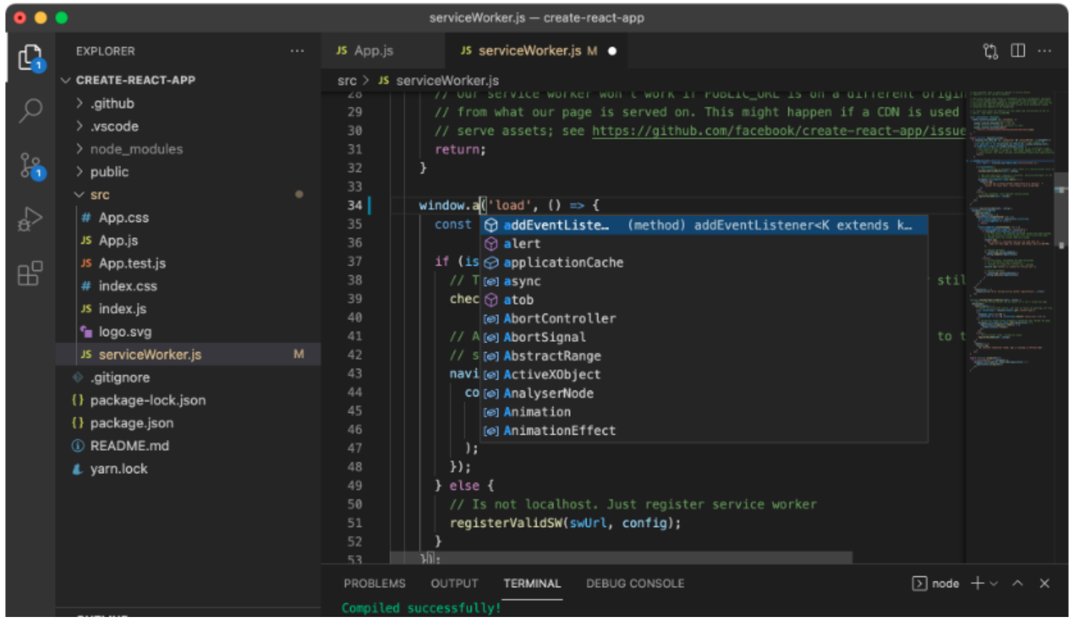
Visual Studio Code - Open Source ("Code - OSS")

feature-request issues 2.9k open | bug issues 2.4k open | chat on gitter

The Repository

This repository ("`Code - OSS`") is where we (Microsoft) develop the Visual Studio Code product together with the community. Not only do we work on code and issues here, we also publish our roadmap, monthly iteration plans, and our endgame plans. This source code is available to everyone under the standard MIT license.

Visual Studio Code

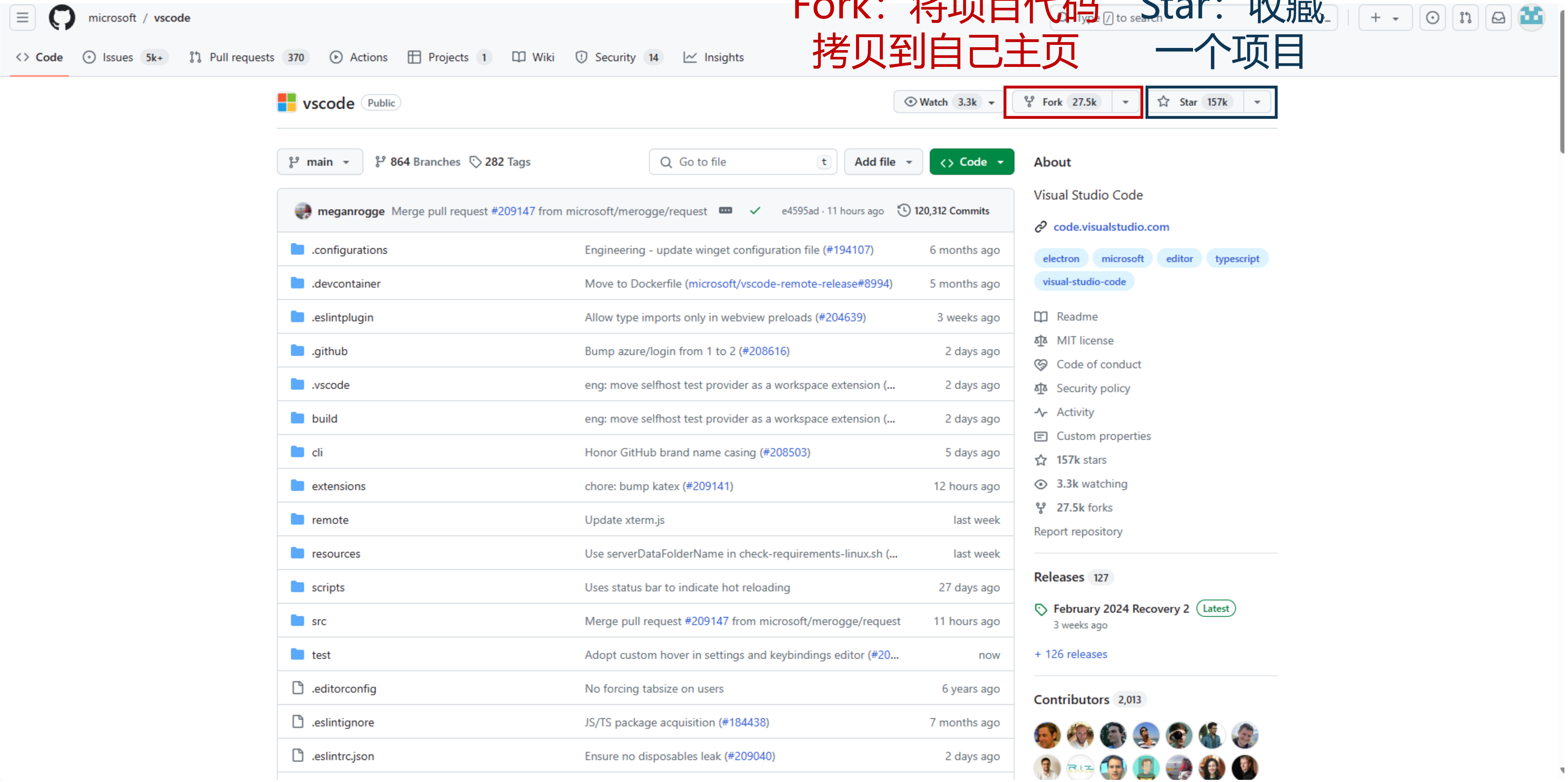


microsoft/vscode: Visual Studio Code (github.com)

GitHub简介

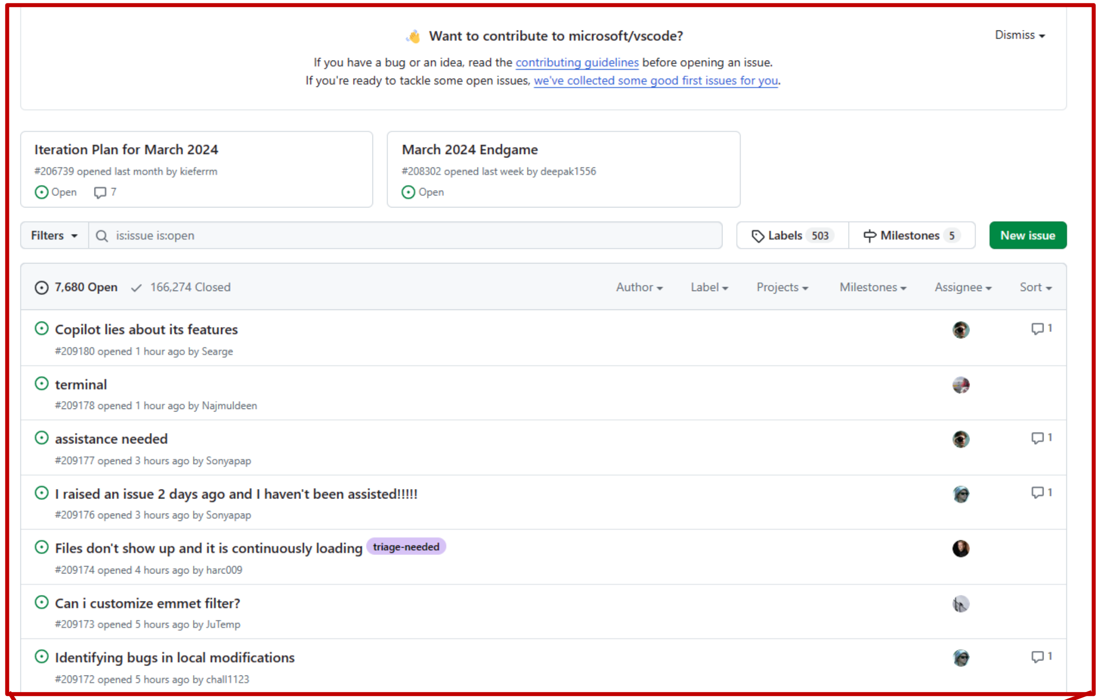
Fork：将项目代码拷贝到自己主页

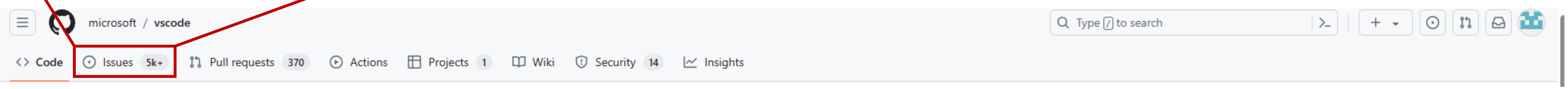
Star：收藏一个项目



[microsoft/vscode: Visual Studio Code (github.com)](https://github.com/microsoft/vscode)

GitHub简介

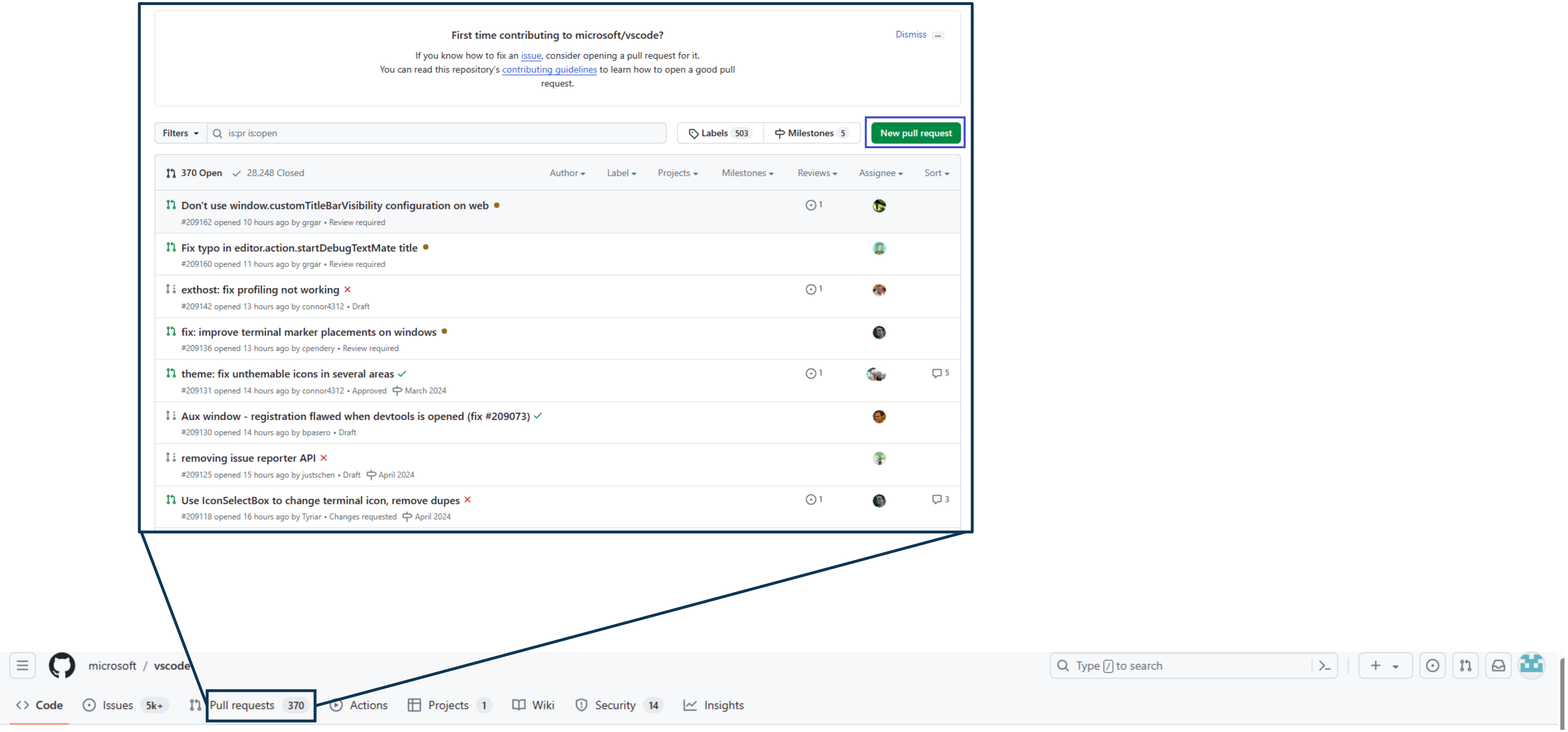




给项目作者提问题，报告bug

[microsoft/vscode: Visual Studio Code (github.com)]

GitHub简介

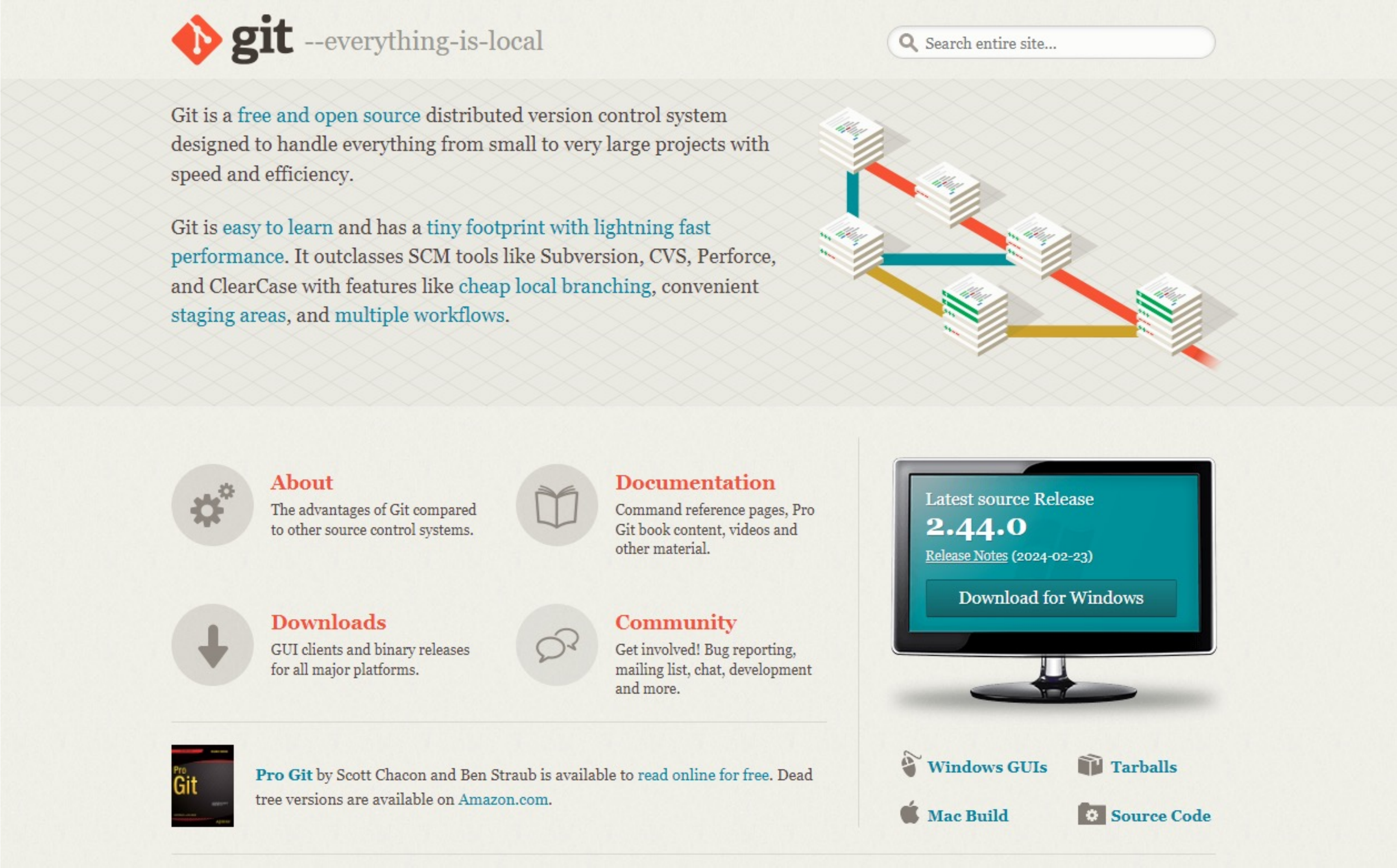


将自己开发的代码合并到主分支或开发分支

[microsoft/vscode: Visual Studio Code (github.com)]

- ❑ GitHub简介
- ❑ **Git基本操作**
- ❑ 协作和贡献代码

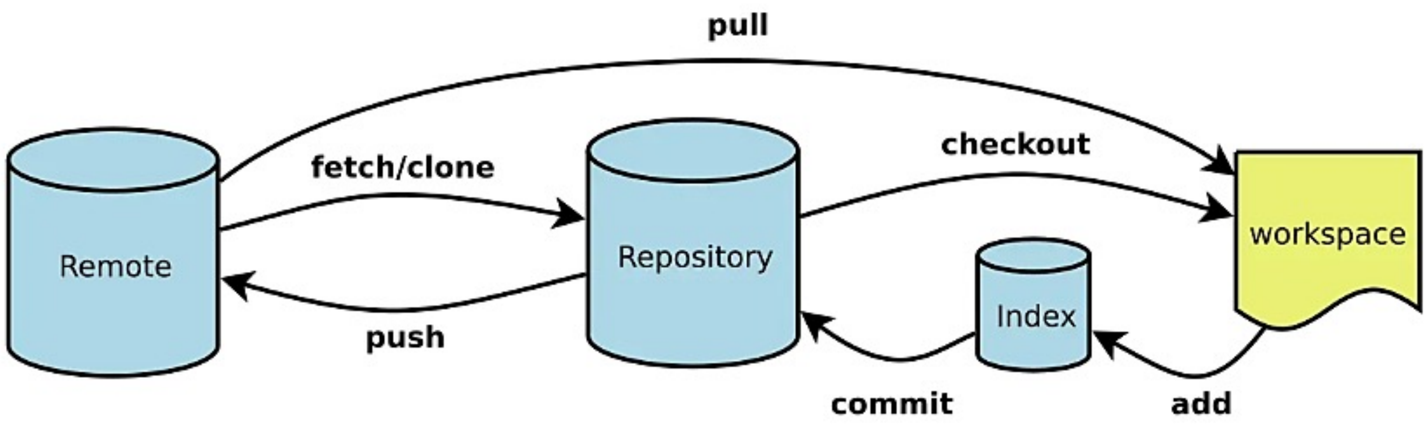
Git基本操作

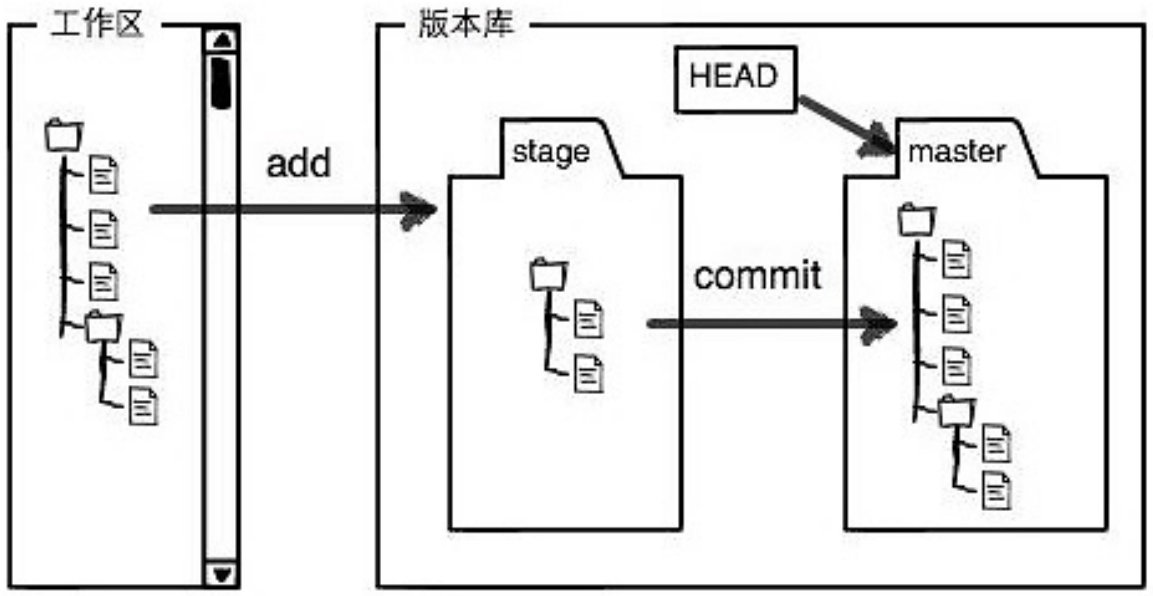


Git (git-scm.com)

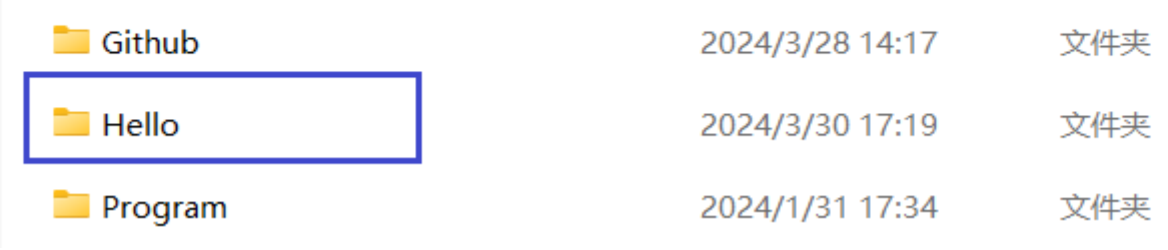
Git基本操作

- **远程仓库（Remote）**
  - 例如GitHub
- **仓库（Repository）**
  - 仓库是一个储存你项目代码历史记录的地方。它包括了所有的提交记录，分支等。
- **工作区（Workspace）**
  - 在计算机上看到的文件和目录，你可以在这里编辑文件
- **暂存区（Index）**
  - 存储了下一次提交即将包含的文件列表和内容快照

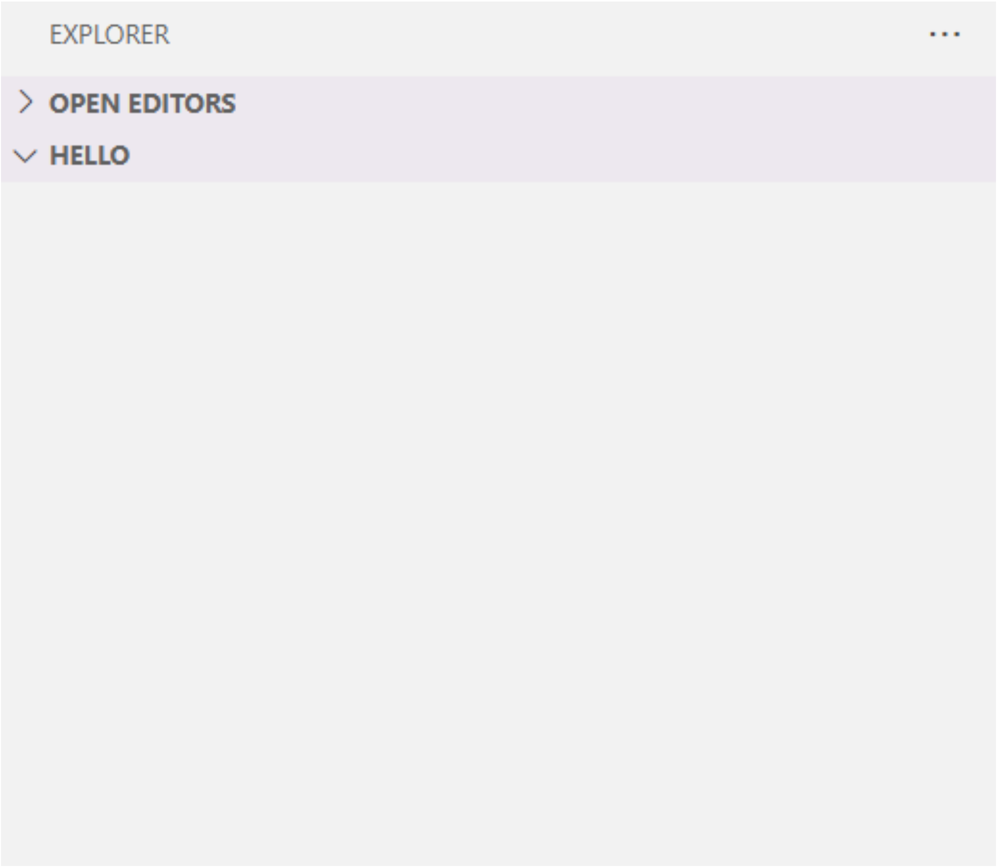




Git基本操作



在本地建立一个文件夹

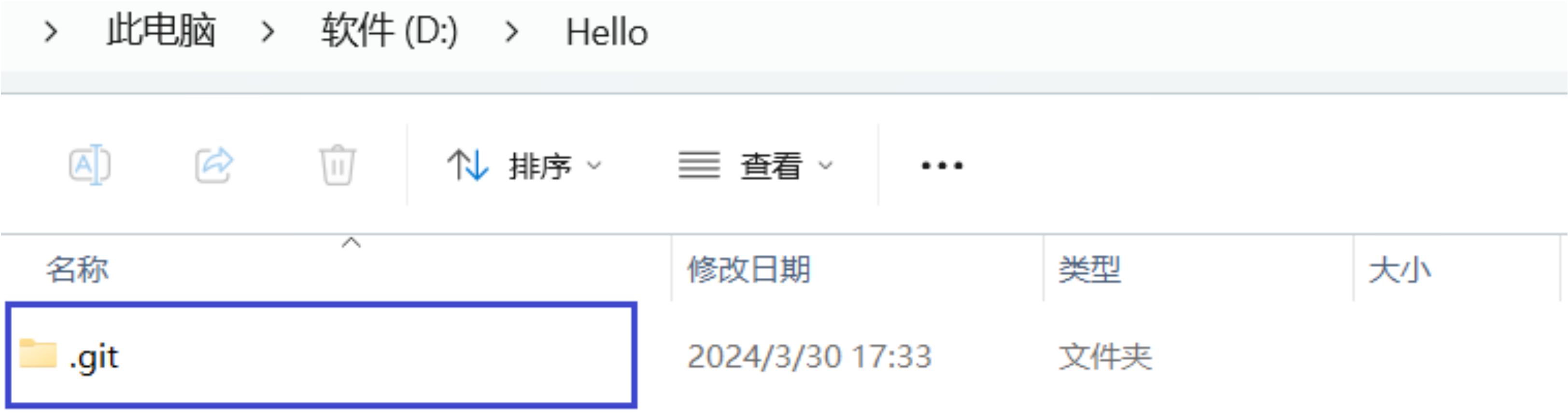


使用VS Code打开

Git基本操作

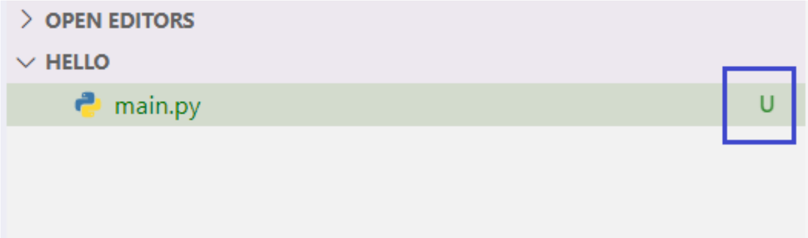
```
# 初始化当前仓库为git仓库
git init
```

```
PS D:\Hello> git init
Initialized empty Git repository in D:/Hello/.git/
PS D:\Hello>
```



Git基本操作

```
# 标记将要提交的文件
git add <文件路径>
```




```
# 显示当前工作目录下的提交文件状态
git status
```

```
PS D:\Hello> git status
On branch new-feature

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        main.py

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Hello>
```

git add .\main.py

```
PS D:\Hello> git add .\main.py
PS D:\Hello>
PS D:\Hello> git status
On branch new-feature

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.py
```

Git基本操作

```
# 创建一个提交并提供提交信息
git commit -m "提交信息"
```

```
# 显示提交历史
git log
```

```
PS D:\Hello> git commit -m "initialize the repo"
[master (root-commit) 917150e] initialize the repo
 1 file changed, 1 insertion(+)
 create mode 100644 main.py
```

```
PS D:\Hello> git status
On branch master
nothing to commit, working tree clean
```

```
PS D:\Hello> git log
commit 917150e792ff4318beb716d0b6b94cf222d68005 (HEAD -> master)
Author: Chen Yiyang <2522570758@qq.com>
Date:   Sat Mar 30 18:01:49 2024 +0800
```

Git基本操作

- 我们现在添加两个文件，分别实现两个函数add和mul
- 然后修改main.py

```
add.py    U
main.py   M
mul.py    U
```

修改main.py

```
main.py M ×   add.py U   mul.py U
main.py > ...
from add import add
from mul import mul

print('hello world')

a = 3
b = 4
ans1 = add(a, b)
ans2 = mul(a, b)
print(f'a+b={ans1}, a*b={ans2}')
```

添加mul.py

```
mul.py > mul
def mul(a, b):
    return a * b
```

添加add.py

```
add.py > add
def add(a, b):
    return a + b
```

```
PS D:\Hello> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        add.py
        mul.py

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\Hello>
```

```
# 将当前文件夹下所有文件标记为提交
git add .
```

```
PS D:\Hello> git add .
PS D:\Hello> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   add.py
        modified:   main.py
        new file:   mul.py
```

Git基本操作

提交本次修改

```
PS D:\Hello> git commit -m "add 2 functions, and modify main.py"
[master 2ca773e] add 2 functions, and modify main.py
 3 files changed, 14 insertions(+), 1 deletion(-)
 create mode 100644 add.py
 create mode 100644 mul.py
```

查看提交日志

```
PS D:\Hello> git log
commit 2ca773e66064eee78398c6c3b9e81ebc537b37ec (HEAD -> master)
Author: Chen Yiyang <2522570758@qq.com>
Date:   Sat Mar 30 18:18:13 2024 +0800

    add 2 functions, and modify main.py

commit 917150e792ff4318beb716d0b6b94cf222d68005
Author: Chen Yiyang <2522570758@qq.com>
Date:   Sat Mar 30 18:01:49 2024 +0800

    initialize the repo
```

Git基本操作

□ 创建分支

```
# 查看分支状态
git branch

# 创建一个分支
git branch 分支名称

# 切换分支
git checkout 分支名称

# 创建一个分支并立刻切换到此分支
git checkout -b 分支名称
```

```
PS D:\Hello> git checkout -b feature
Switched to a new branch 'feature'
```

```
PS D:\Hello> git branch
* feature
  master
```

这里我们创建了一个新分支，名称是feature

```
add.py
main.py    M
mul.py
sub.py     U
```

```
sub.py > sub
def sub(a, b):
    return a - b
```

```
main.py > ...
from add import add
from mul import mul
from sub import sub

print('hello world')

a = 3
b = 4
ans1 = add(a, b)
ans2 = mul(a, b)
ans3 = sub(a, b)
print(f'a+b={ans1}, a*b={ans2}, a-b={ans3}')
```

```
git add
git commit
```

```
PS D:\Hello>
PS D:\Hello> git add .
PS D:\Hello>
PS D:\Hello> git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.py
        new file:   sub.py

PS D:\Hello> git commit -m "add sub function"
[feature 9cc88ff] add sub function
 2 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 sub.py
```

在新分支上添加文件sub.py，修改main.py，然后提交

Git基本操作

- 合并分支

```
# 合并分支
# （需要首先切换到需要被合并的分支上）
git merge 分支名称
```

切换到主分支

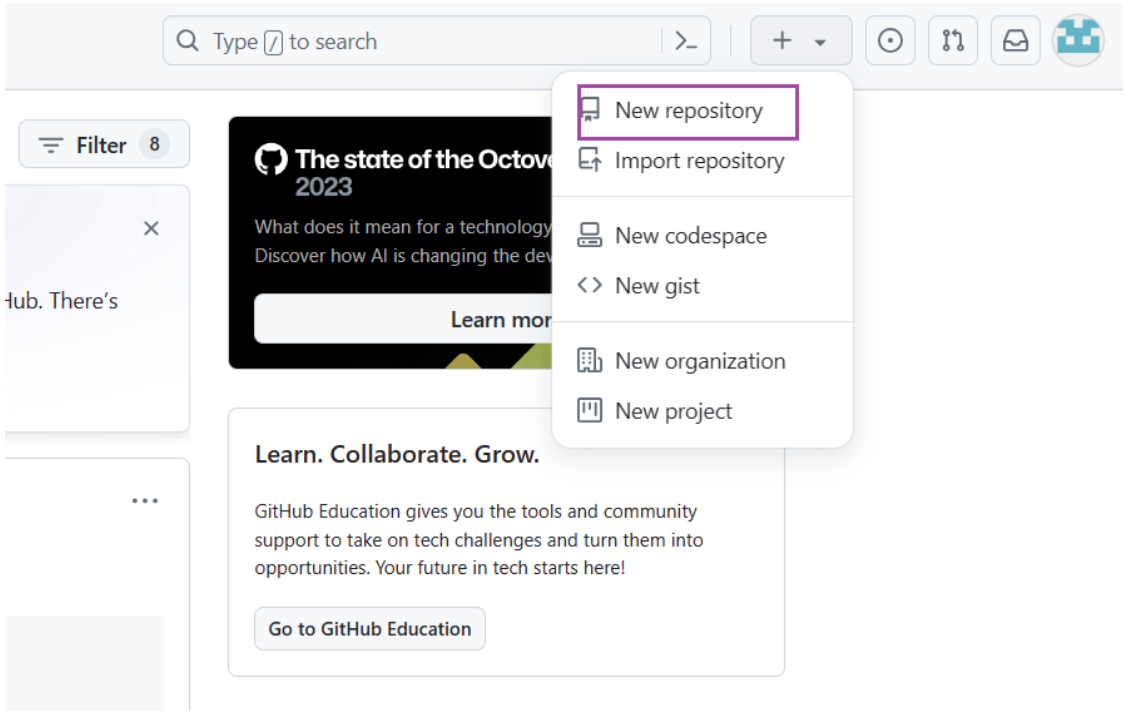
```
PS D:\Hello> git checkout master
Switched to branch 'master'
```

合并到主分支上

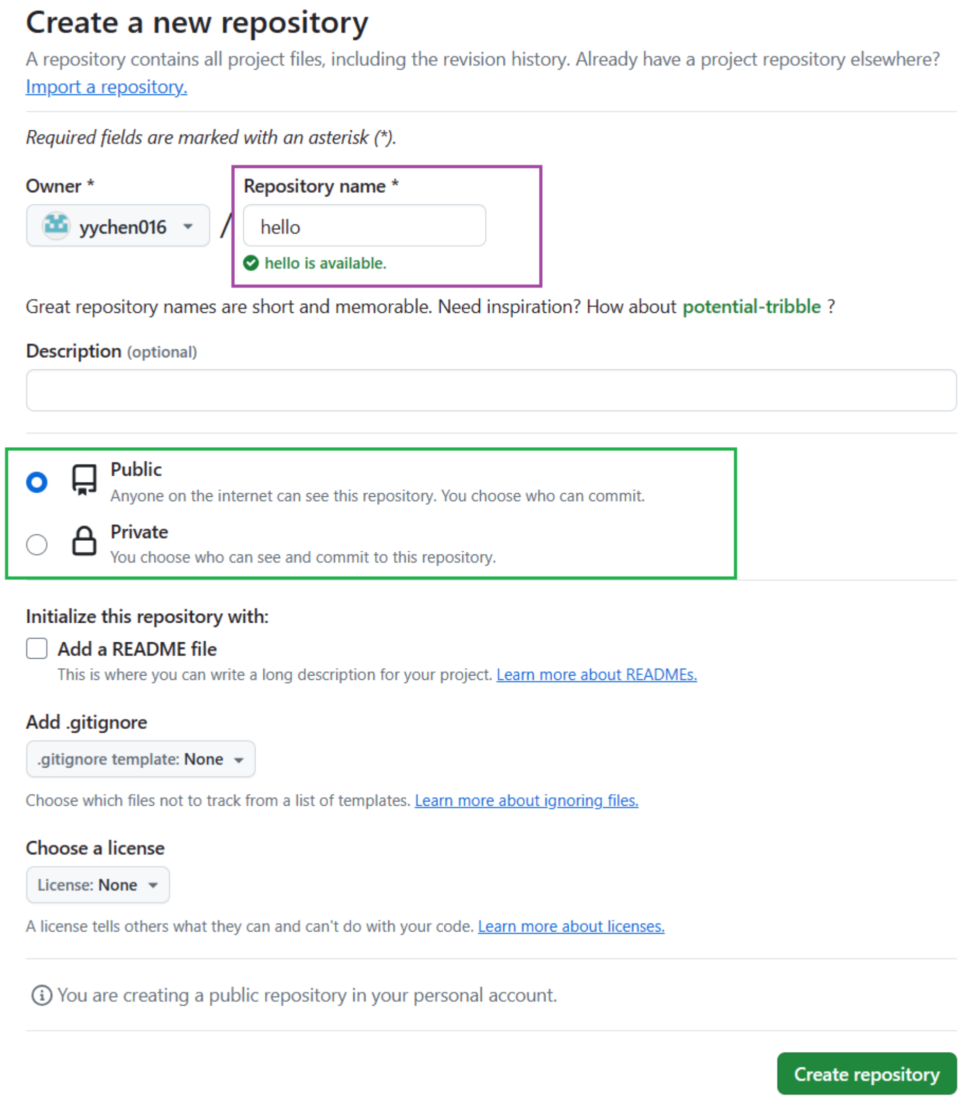
```
PS D:\Hello> git merge feature
Updating 2ca773e..9cc88ff
Fast-forward
 main.py | 4 +++-
 sub.py  | 2 ++
 2 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 sub.py
```

Git基本操作

- 将代码提交到Github上

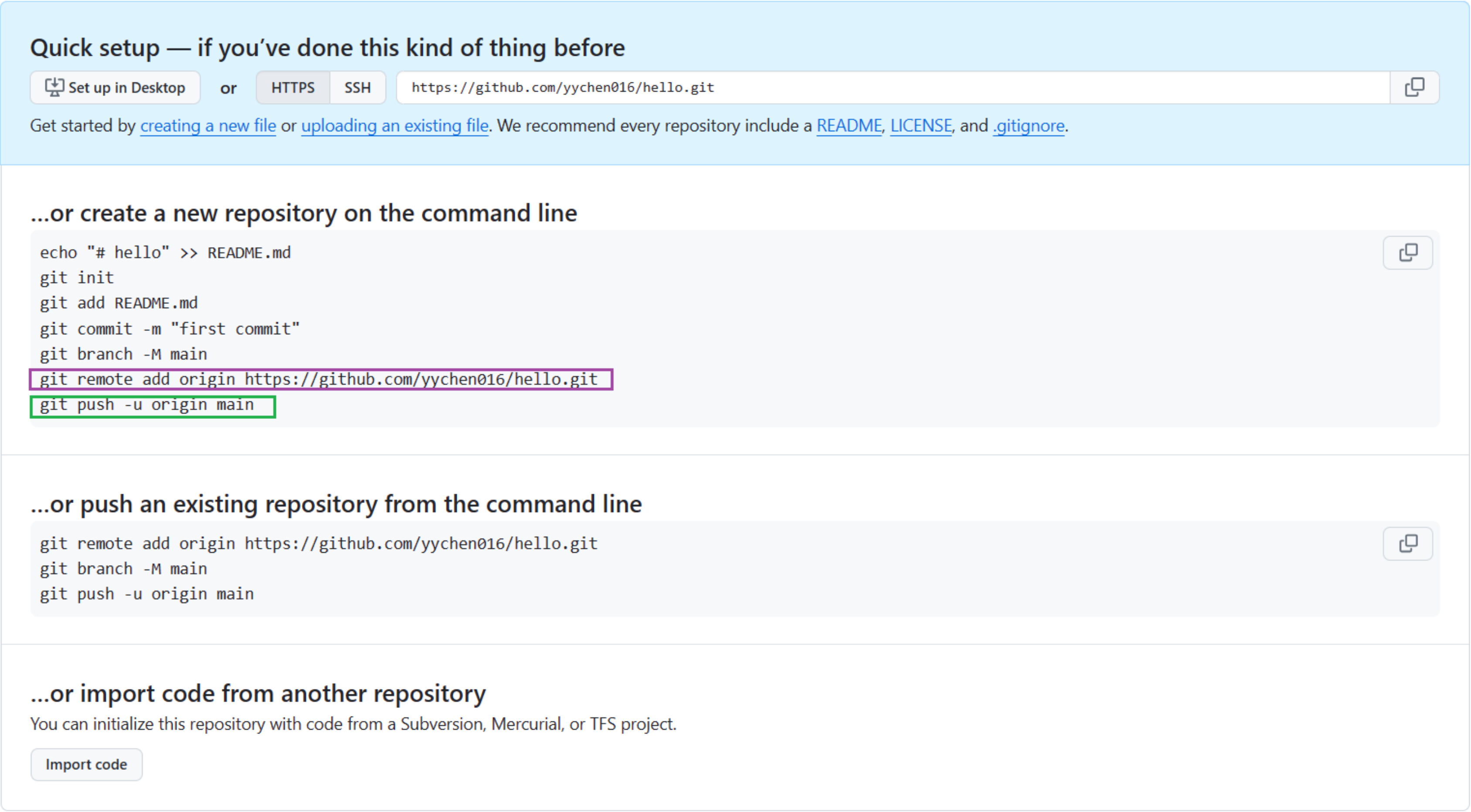


首先创建远程仓库



Git基本操作

□ 关联本地仓库与远程仓库



NOTE：这里它首先创建了一个分支main，我们可以直接使用本地已经有的分支master上传

Git基本操作

◘ 关联本地仓库与远程仓库

在你的本地Git仓库中添加一个新的远程仓库引用

```
PS D:\Hello> git remote add origin https://github.com/yychen016/hello.git
PS D:\Hello>
PS D:\Hello> git remote
origin
```

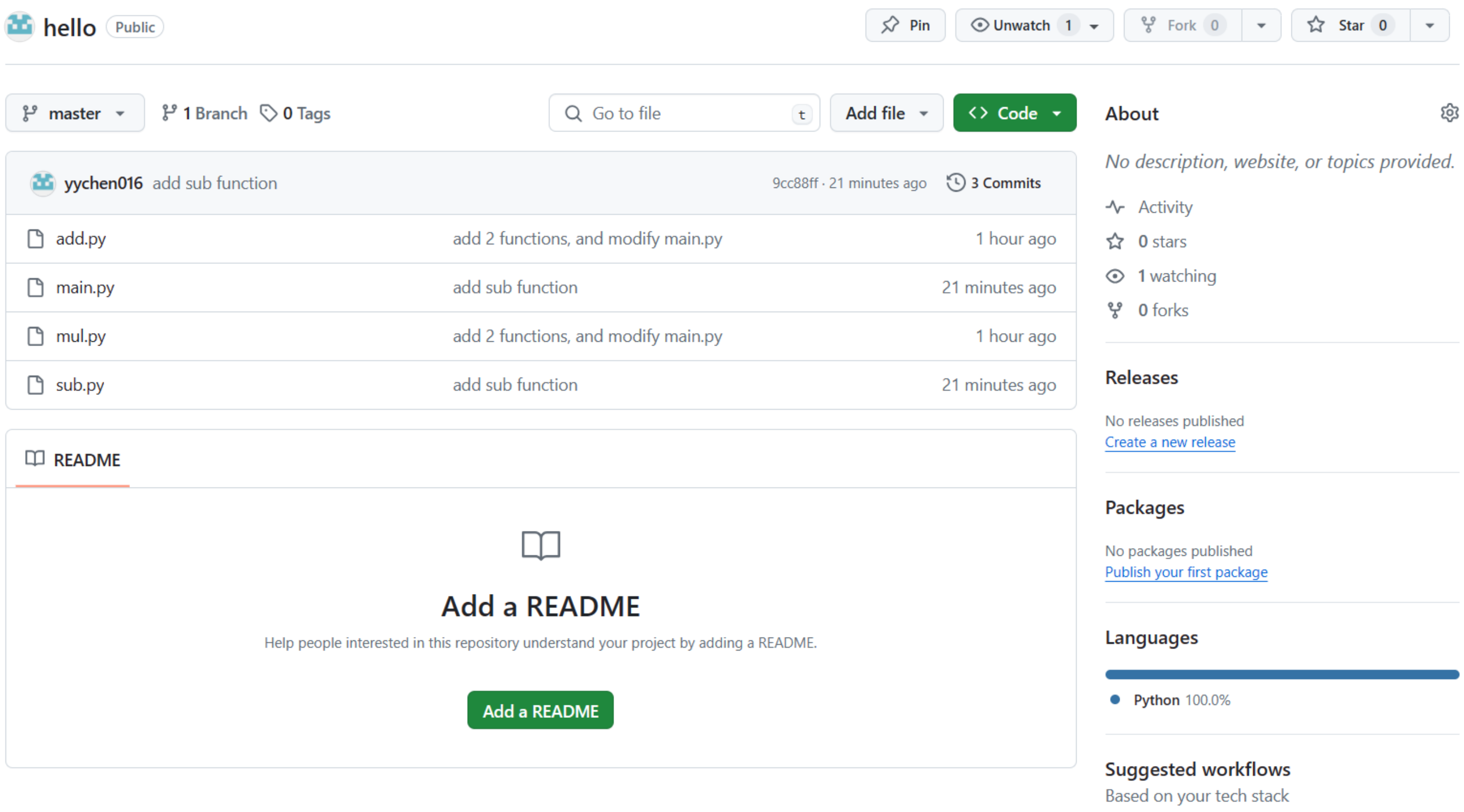
git push –u origin master

将你的本地master分支的更改推送到远程仓库

```
PS D:\Hello> git push -u origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (12/12), 1.04 KiB | 533.00 KiB/s, done
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/yychen016/hello.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

这里的-u参数会将你的本地分支与远程分支关联起来，以后你只需要运行git push或git pull就可以同步更改。

Git基本操作



代码已经成功上传到Github上

- ❑ GitHub简介
- ❑ Git基本操作
- ❑ **协作和贡献**

协作和贡献

◻ 如何参与别人的项目?

1. 找到项目并Fork到自己仓库
   - 在项目页面的右上角，点击"Fork"按钮。这会创建该项目的一个副本到你的GitHub账户下，你将拥有这个副本的完全控制权

2. 克隆仓库
   - 使用Git将Fork后的项目克隆到你的本地计算机上。在你的仓库页面中找到"Clone or download"按钮，复制提供的URL。
   - 使用：git clone <仓库URL>

3. 本地修改代码、开发新功能

协作和贡献

如何参与别人的项目？

4. 推送修改到Github
   - `git push origin <分支名>`

5. 提交Pull Request（PR）
   - 在GitHub上到你Fork的仓库页面，选择"Pull requests"标签页，然后点击"New pull request"。
   - 选择你刚推送的分支，填写PR的详细信息，解释你的更改及其原因。
   - 提交PR

6. 等待反馈
   - 项目维护者会查看你的PR，可能会提出更改请求。根据反馈进行相应的修改，并在PR评论中进行讨论。如果你的PR被接受，维护者会将它合并到原项目中。

Git指令总结

```
# 初始化当前仓库为git仓库
git init

# 标记将要提交的文件
git add <文件路径>

# 显示当前工作目录下的提交文件状态
git status

# 创建一个提交并提供提交信息
git commit -m "提交信息"

# 显示提交历史
git log

# 查看分支状态
git branch
```

```
# 创建一个分支
git branch 分支名称

# 切换分支
git checkout 分支名称

# 创建一个分支并立刻切换到此分支
git checkout -b 分支名称

# 合并分支
# （需要首先切换到需要被合并的分支上）
git merge 分支名称

# 向远程仓库推送
git push

# 向远程仓库拉取
git pull
```

更多详细内容可以参考：[Git教程 - 廖雪峰的官方网站 (liaoxuefeng.com)]

谢 谢!