《范数，K近邻与支持向量机》

崔志勇

交通科学与工程学院

2024年5月

向量的范数

向量的范数是一个将向量映射到标量值的函数。

- 范数是衡量向量大小的一种度量方式

向量的范数 f 应该满足以下性质：

- **缩放性**：对于任意向量 x 和任意标量 α，范数满足

$$f(\alpha\mathbf{x}) = |\alpha| f(\mathbf{x})$$

- **三角不等式**：对于任意的两个向量 x 和 y，范数满足

$$f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$$

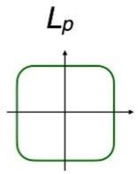
- **非负性**：对于任意向量 x，范数满足

$$f(\mathbf{x}) \geq 0$$

向量的范数

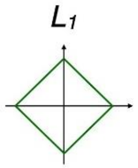
向量的一般 ℓp 范数是通过以下方式获得的：

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$



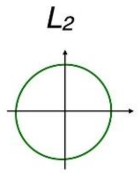
当 p=1时，我们有 ℓ1 范数。这种范数使用元素的绝对值来计算，能够区分零元素和非零元素。

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$



当 p=2 时，我们得到的是 ℓ2 范数，也称为欧几里得范数或欧氏距离。它是最常用的范数，因为它与我们在几何中直观理解的直线距离相对应。

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$

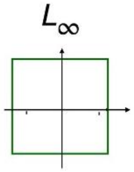


ℓ2 范数通常可以省略下标 2，直接表示为 $\|\mathbf{x}\|$

向量的范数

当 $p=\infty$ 时，我们得到的是 $\ell\infty$ 范数。它通常被称为无穷范数或最大范数。 它的作用是找出向量中绝对值最大的那个元素。

$$\|\mathbf{x}\|_\infty = \max_i |x_i|$$



$\ell 0$ 范数输出的是非零元素的数量。 它不是一个 ℓp 范数，实际上它也不是一种真正的范数函数（它被称为范数是不准确的）。

正则化

λ= 正则化强度(超参数)

$$L = -\frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2 + \boxed{\lambda R(W)}$$

以下是几种常见的正则化方法:

L2 正则化（也称为岭回归）：

$$R(W) = \|w\|_2$$

L1 正则化（也称为Lasso回归）：

$$R(W) = \|w\|_1$$

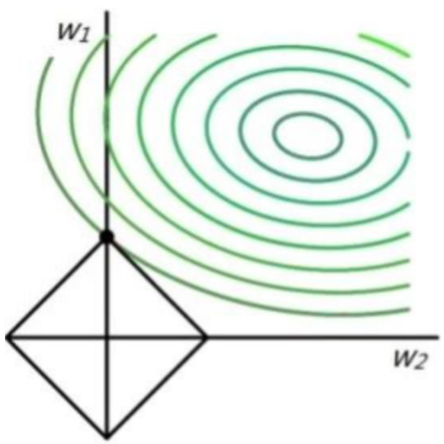
Elastic Net（弹性网络，L1 和 L2 的结合）：

$$R(W) = \alpha\|w\|_2 + \|w\|_1$$

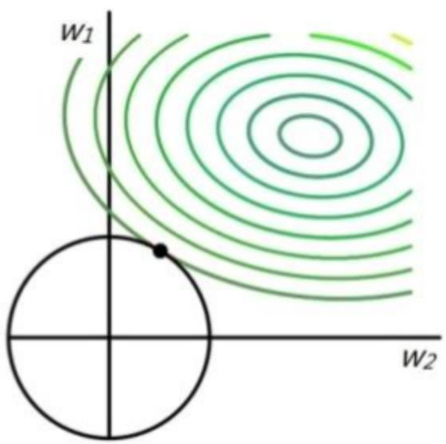
正则化

为什么L1正则化产生稀疏的权值，L2正则化产生平滑的权值？

- 从几何位置关系的角度来看权值



(a) L1 Regulariztion



(b) L2 Regularization

•L1在和每个坐标轴相交的地方都有“角”出现
•目标函数的测地线除非位置摆得非常好，大部分时候都会在角的地方相交。注意到在角的位置就会产生稀疏性，例如图中的相交点就有w1=0，

•L2因为没有角，第一次相交的地方出现在具有稀疏性的位置的概率就变得非常小

KNN

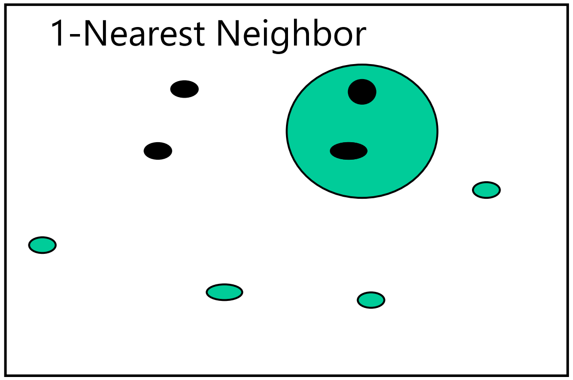
KNN：K-Nearest Neighbor （K最近邻算法）

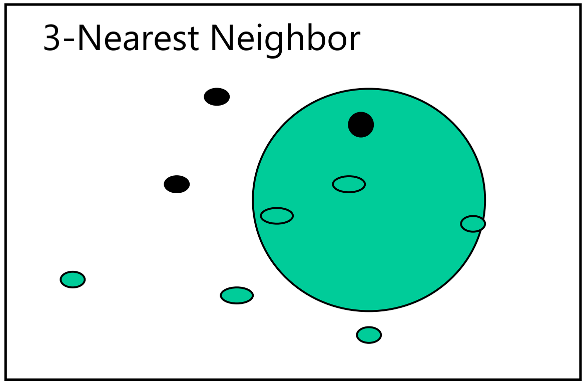
k-NN是一种基于实例的学习，或者称为惰性学习，其中函数只在局部近似，所有计算都推迟到分类进行时。k-NN算法是最简单的机器学习算法之一。

KNN

- k-NN分类，输出的是类的隶属度。一个对象由其邻居的类别投票进行分类，对象被分配到其k个最近邻居中最常见的类（k是一个正整数，通常很小）。如果k=1，那么对象就被简单地分配给单个最近邻居的类。

- k-NN回归, 输出是对象的属性值。该值是其k个最近邻值的平均值。





KNN

- 任意的实例可以表示为$(x_1, x_2, \dots, x_n)$
- 两个实例之间的欧几里得距离表示为：

$$\text{dist}\,(x_i, x_j) = \sqrt{(x_i - x_j)^2}$$

- 若实例有K维特征：

$$\text{dist}\,(x_i, x_j) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(x_i^k - x_j^k)^2}$$

连续目标函数值

- K个最近邻训练样本的均值
- K个最近邻训练样本的加权平均值（例如最常见的加权为1/d, d代表了实例间的距离）。

KNN总结

- 所有实例都对应于n维欧氏空间中的点
- 高计算负担
- 通过比较不同点的特征向量进行分类
- 目标函数可以是离散函数或实值函数
- 高噪声训练数据和复杂目标函数的高效归纳推理方法

SVM支持向量机

支持向量机(SVMs，有时也被叫做支持向量网络)是监督学习模型，包括了对应的算法，可以用于分类和回归分析。

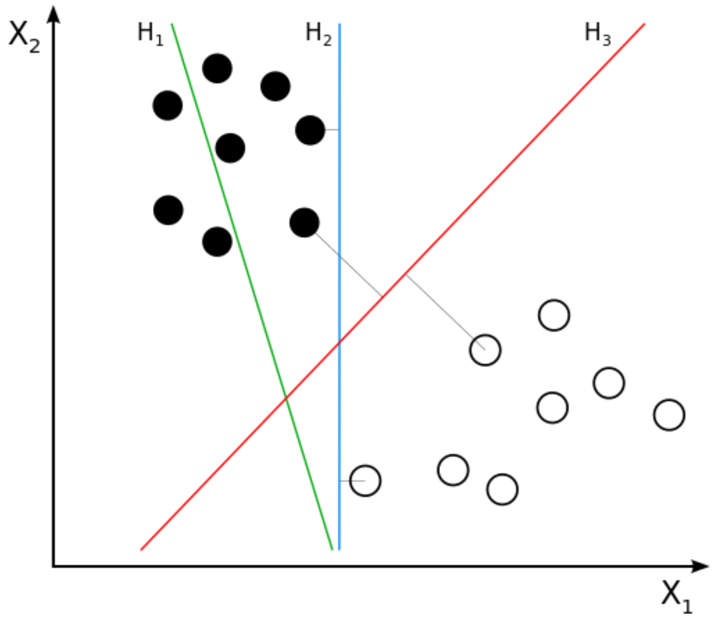
支持向量机模型是将实例表示为空间中的点，通过映射使单独类别的实例**被一个尽可能宽的间隔完全分割。**

然后，新的例子被映射到同一个空间，并根据它们落在差距的哪一边被预测为属于一个类别。

SVM支持向量机

假设我们有一些二维观测数据，即属于两个类别的点对 ($x1$,$x2$)。

我们可以在平面上将它们绘制为黑色和白色的点。在这种情况下，每个点代表一个观测值，其坐标由 $x1$ 和 $x2$ 确定，颜色则表示它们所属的类别。黑色点代表一个类别，而白色点代表另一个类别。

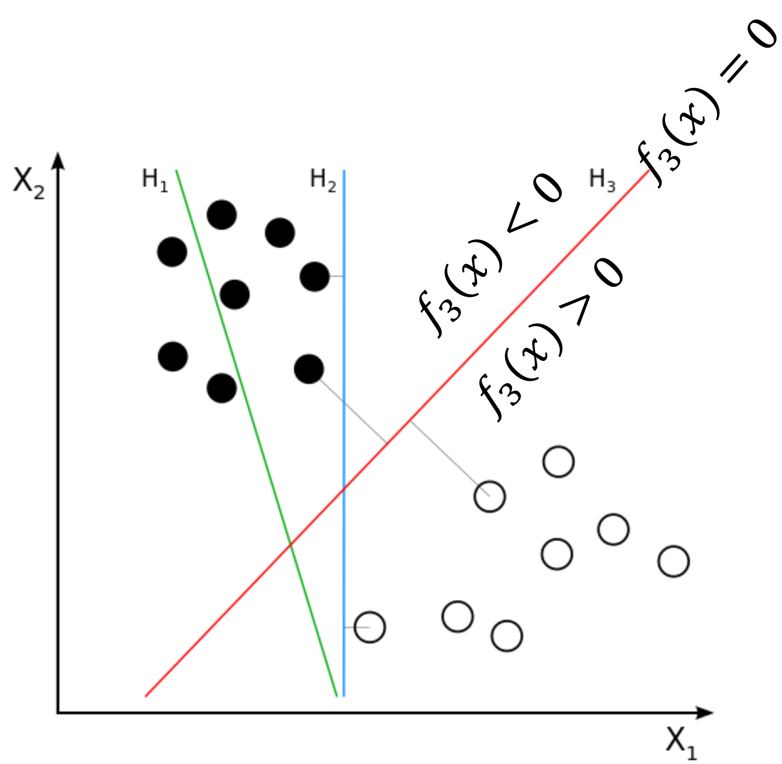


SVM支持向量机

我们可以构建一个线性分类器，它使用权重向量 **w**=(w0,w1,w2) 和决策函数来区分不同的类别。这里的决策函数定义如下：

$$f(x) = w_0 + w_1x_1 + w_2x_2 > 0$$

决策边界是满足 $f(x)$=0的点的集合，也就是说，这是**将两类数据分开的边界。**在图中，有三条不同的线展示了三种不同的决策边界，分别对应于三个不同的权重向量 **w**j，其中 j∈{1,2,3}，并且分别对应于分类器 $H1$,$H2$,$H3$

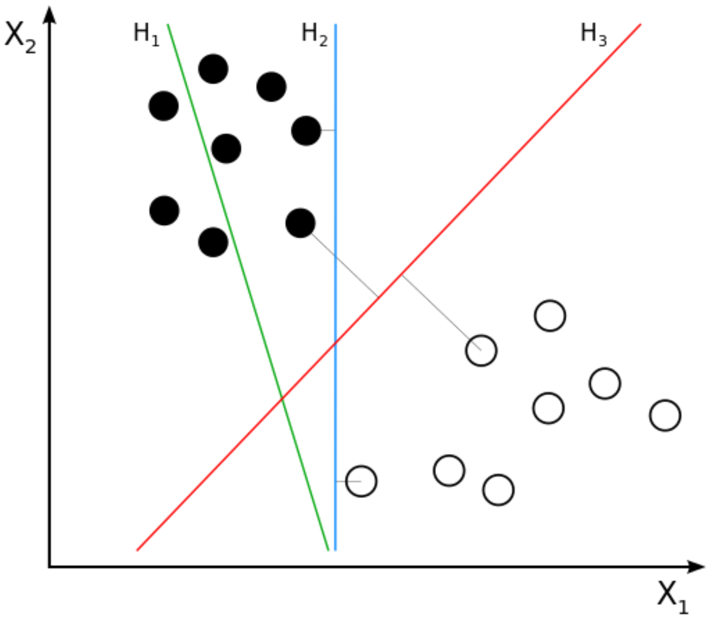


SVM支持向量机

*H*1 没有正确地对数据进行分类，而 *H*2 和 *H*3 都做到了。

但是 *H*2 和 *H*3 所定义的分类器并不是同样优秀的

哪个分类器（*H*2 还是 *H*3）的边界能提供更好的分类效果，以及为什么？



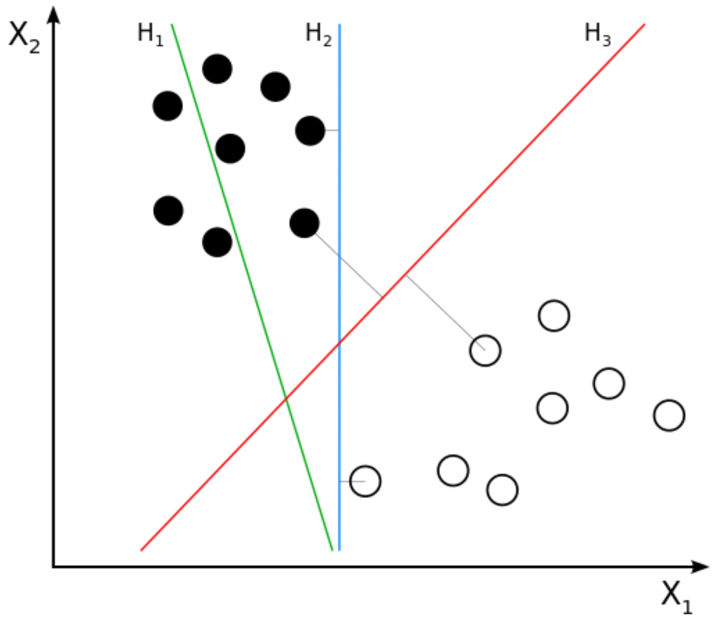
SVM支持向量机

*H*1 没有正确地对数据进行分类，而 *H*2 和 *H*3 都做到了。

但是 *H*2 和 *H*3 所定义的分类器并不是同样优秀的

哪个分类器（*H*2 还是 *H*3）的边界能提供更好的分类效果，以及为什么？

***H*3 拥有更大的间隔**（灰色线），因此**更有可能**正确地对新数据进行分类。

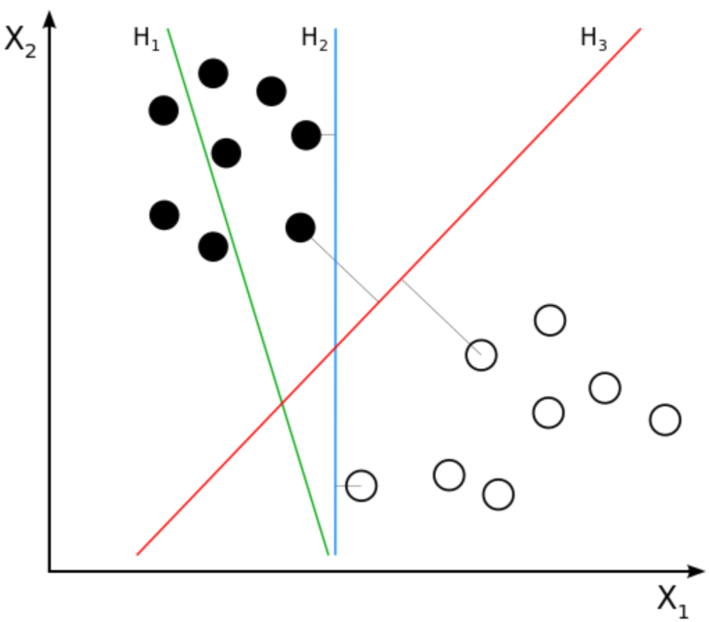


SVM支持向量机

H3 拥有更大的**间隔**（灰色线），因此更有可能正确地对新数据进行分类。

新数据点要想被错误分类，**至少需要与正确分类的点保持跟间隔一样长的距离。**

较大的间隔意味着这些点必须远离现有的数据点，这种情况发生的可能性较小。

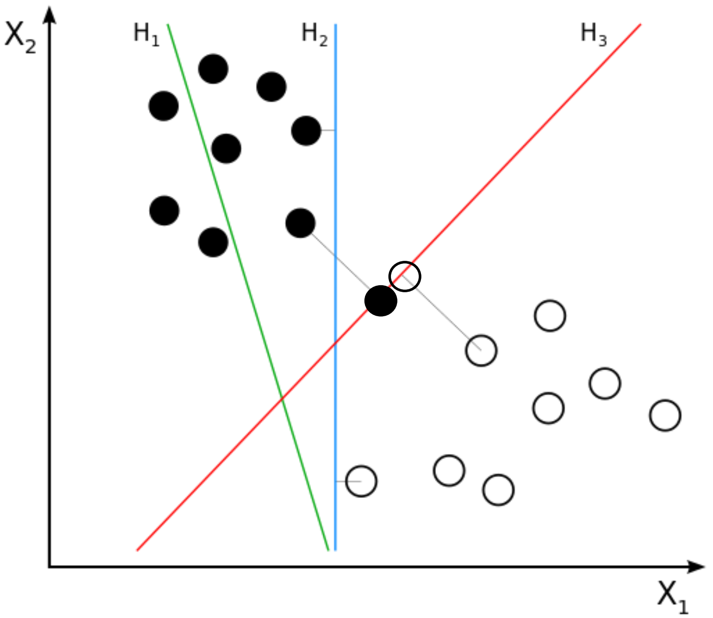


SVM支持向量机

H3 拥有更大的**间隔**（灰色线），因此更有可能正确地对新数据进行分类。

新数据点要想被错误分类，**至少需要与正确分类的点保持跟间隔一样长的距离。**

较大的间隔意味着这些点必须远离现有的数据点，这种情况发生的可能性较小。

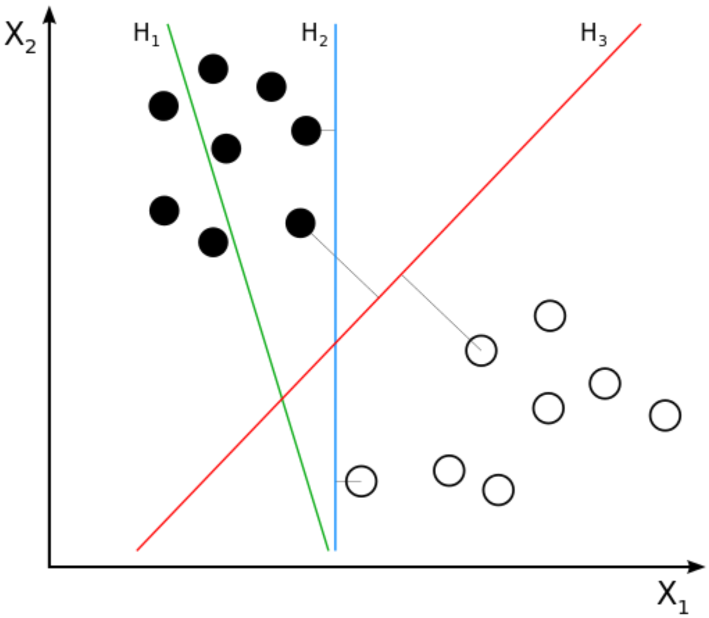


SVM支持向量机

*H*3 拥有更大的**间隔**（灰色线），因此更有可能正确地对新数据进行分类。

新数据点要想被错误分类，**至少需要与正确分类的点保持跟间隔一样长的距离。**

较小的间隔意味着这些点可能紧邻当前的数据点，这种情况发生的可能性较大。

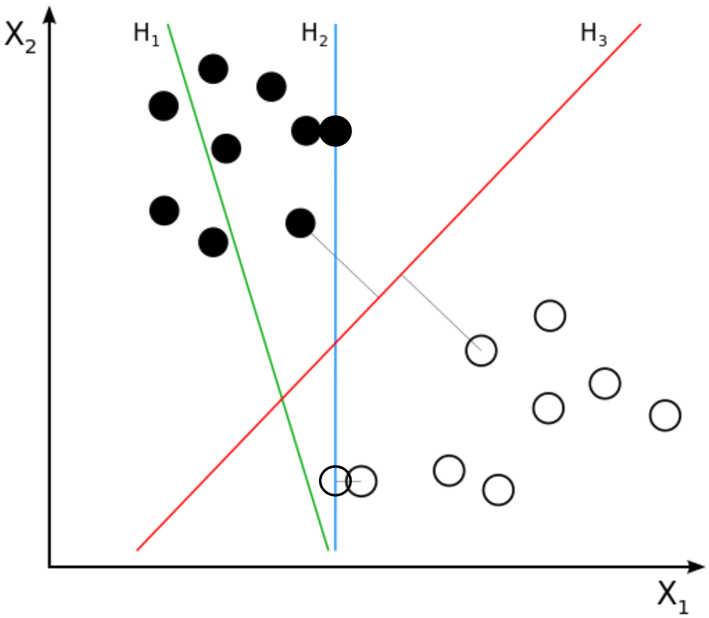


SVM支持向量机

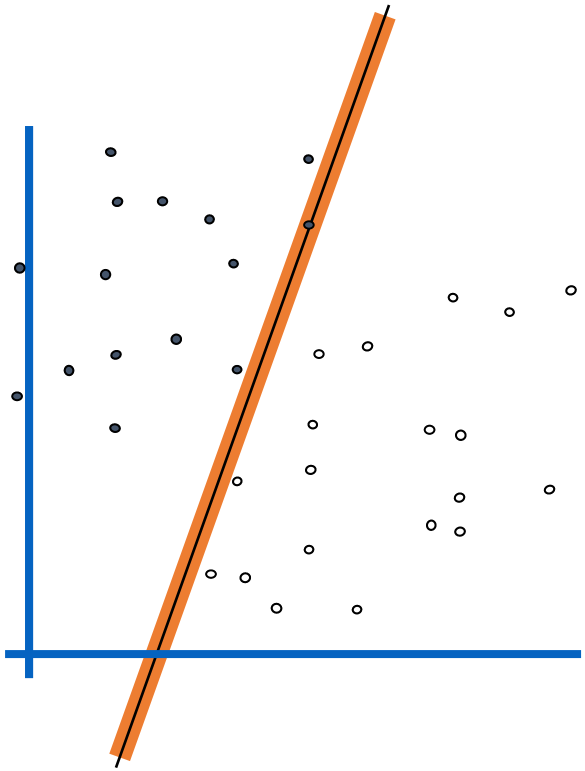
*H*3 拥有更大的**间隔**（灰色线），因此更有可能正确地对新数据进行分类。

新数据点要想被错误分类，**至少需要与正确分类的点保持跟间隔一样长的距离。**

较小的间隔意味着这些点可能紧邻当前的数据点，这种情况发生的可能性较大。



分类边距



$f(\mathbf{x}) = sign(\mathbf{w}\ \mathbf{x} + b)$

是符号函数，将自变量，进一步映射到的输出类别 {+1,−1} 上

将线性分类器的**边距**定义为：

边界在到达数据点之前可以增加的宽度。

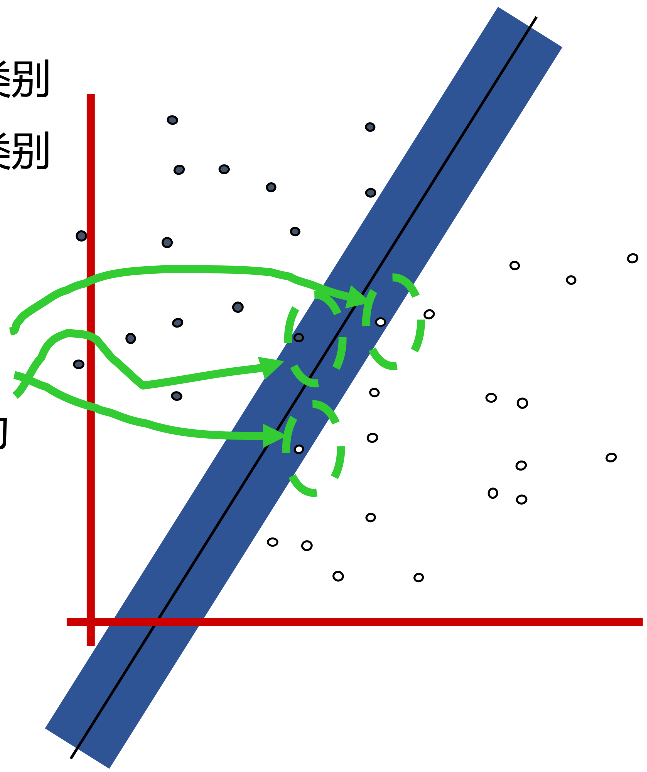
SVM支持向量机

最大化分类边距

- 代表正类别
- 代表负类别

支持向量

为边距最大处的数据点



最大化线性分类器的边距使得线性分类器的**边距最大。**

这是最简单的一种SVM（LSVM）

Linear SVM

单选题 1分

设置

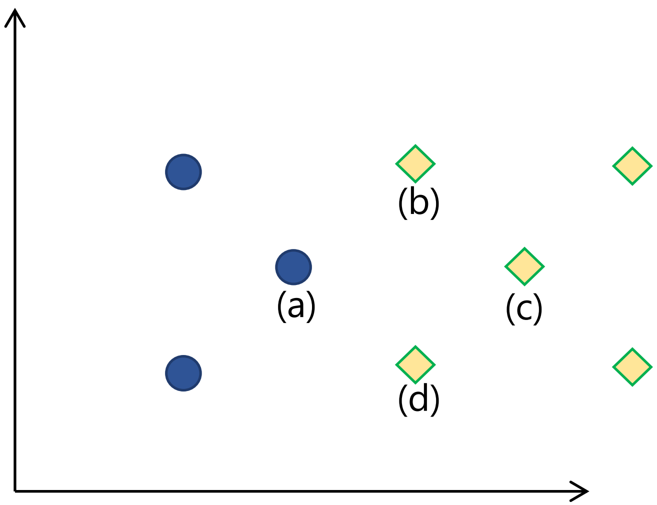
假设有一个线性SVM分类器用来处理二分类问题，移除下图中哪个数据点，决策边界不会变化？

A (a)

B (b)

C (c)

D (d)



提交

SVM支持向量机

SVM Loss

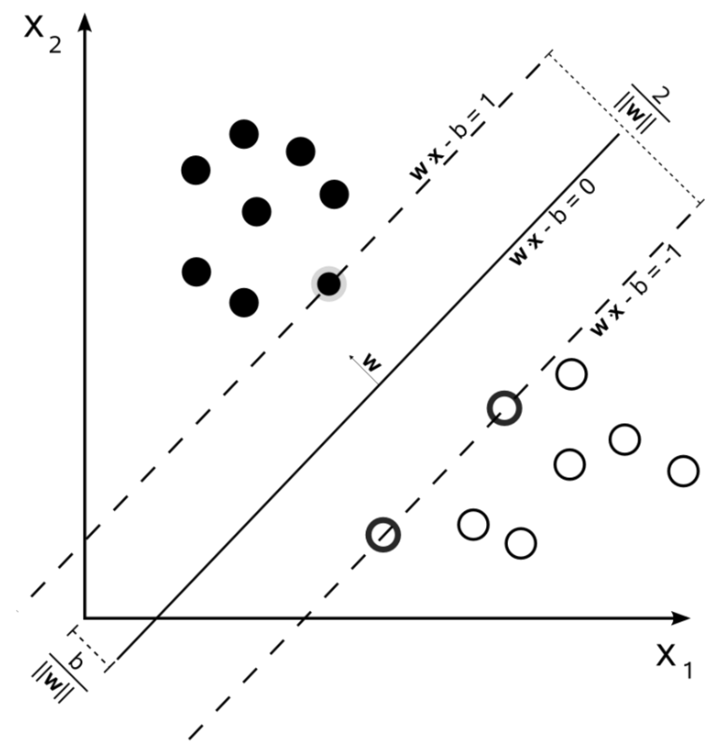
我们希望最大化分类器的间隔，这是通过几个阶段来实现的。

如果权重向量 **w** 的ℓ2范数（长度） $||w||_2 = 1$ ，那么 $f(x) = w^T x + b$就是从点 x 到直线 $f(x) = 0$的距离的有符号度量。

我们可以利用直线 $f(x)$=1和 $f(x)$=−1（见右侧图示）来创建**一个单位长度的间隔**，并设置：

$f(x)$≥1　x∈C

$f(x)$≤−1　x∉C

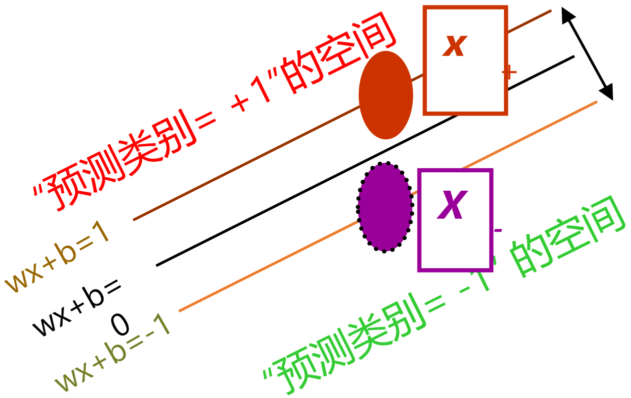


注：点 (x0, y0) 到直线距离 Ax + By + c = 0

Dist = $\frac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}}$

SVM支持向量机

线性SVM数学推导



M=边距大小

$$(x^+ - x^-) \cdot |w| = 2$$

$$(x^+ - x^-) = \frac{2}{|w|}$$

$$\Rightarrow M = \frac{2}{|w|}$$

其中:

$\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$

$\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$

$\boldsymbol{w} \,.\, (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

SVM支持向量机

Hinge Loss（合页损失）

接下来，我们使用一个常见的标记技巧，定义 y 如下：

$$y = \begin{cases} -1 & \text{if} \quad x \notin \mathrm{C} \\ 1 & \text{if} \quad \mathrm{x} \in \mathrm{C} \end{cases}$$

然后，我们将两个不等式简化为一个

$f(x) \geq 1$ for $x \in C$

$f(x) \leq -1$ for $x \notin C$

简化为：

$$yf(x) \geq 1$$

我们希望找到一个损失函数，用来衡量这个约束条件被**违反**了多少：

$$\max(0, 1 - yf(x))$$

这被称为**合页损失（Hinge Loss）**。“Hinge”在这里的意思是“铰链”，暗示损失函数在 $yf(x)$ 大于等于1 时为 0（即没有损失），小于 1 时损失随着差距增大而增大，类似于铰链的开合。

SVM支持向量机

Hinge Loss（合页损失）

接下来，我们使用一个常见的标记技巧，定义 y 如下：

$$y = \begin{cases} -1 & \text{if} \quad x \notin \mathrm{C} \\ 1 & \text{if} \quad \mathrm{x} \in \mathrm{C} \end{cases}$$

然后，我们将两个不等式简化为一个

$f(x) \geq 1$ for $x \in C$

$f(x) \leq -1$ for $x \notin C$

简化为：

$$yf(x) \geq 1$$

我们希望找到一个损失函数，用来衡量这个约束条件被**违反**了多少：

$$\max(0, 1 - yf(x))$$

这被称为**合页损失（Hinge Loss）**。“Hinge”在这里的意思是“铰链”，暗示损失函数在 $yf(x)$ 大于等于1 时为 0（即没有损失），小于 1 时损失随着差距增大而增大，类似于铰链的开合。

SVM支持向量机

Hinge Loss（合页损失）

对于一组数据点，合页损失仅仅是每个点损失的总和：

$$l = \sum_{i=1}^{n} \max(0,1 - y_i f(x_i))$$

到目前为止，我们假设 $f(x) = w^T x + b$且 $||w||_2 = 1$ 。

随着 $\|w\|$ 的增加，合页损失会减少。为什么会这样？

最小化 ℓ 会对促使$\|w\|$ 增加。因此我们添加了一个损失项 $\lambda||w||^2$ ，这有助于减少 $\|w\|$，而不是强制要求$\|w\|$=1：

$$l = \sum_{i=1}^{n} \max(0,1 - y_i f(x_i)) + \lambda||w||^2$$

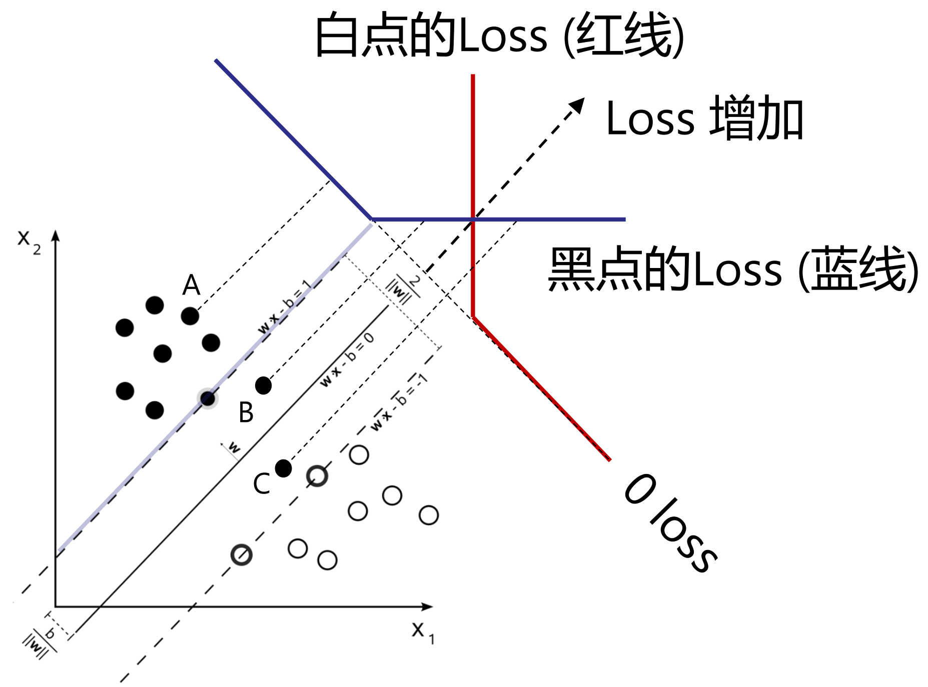
这里的 λ 可以调整，以便使 $\|w\|$接近 1。也就是说，这个分类器既学习间隔也学习边界。这就是**软间隔支持向量机。**

SVM支持向量机

Hinge Loss（合页损失）

我们可以在图表上展示正例和负例的合页损失：

- 正确分类且位于间隔外的点损失为 0（A 类点）
- 正确分类但位于间隔内的点损失为正值（B 类点）
- 损失随着距离间隔的远近而增长
- 错误分类的点有正值损失（C 类点）

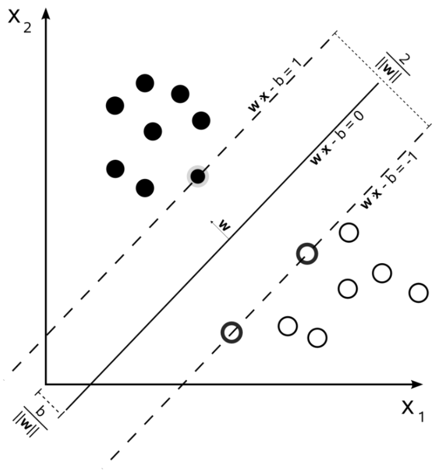


SVM支持向量机

与逻辑回归的比较

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

逻辑回归的（交叉熵）损失对于这些点是如何变化的?



SVM支持向量机

与逻辑回归的比较

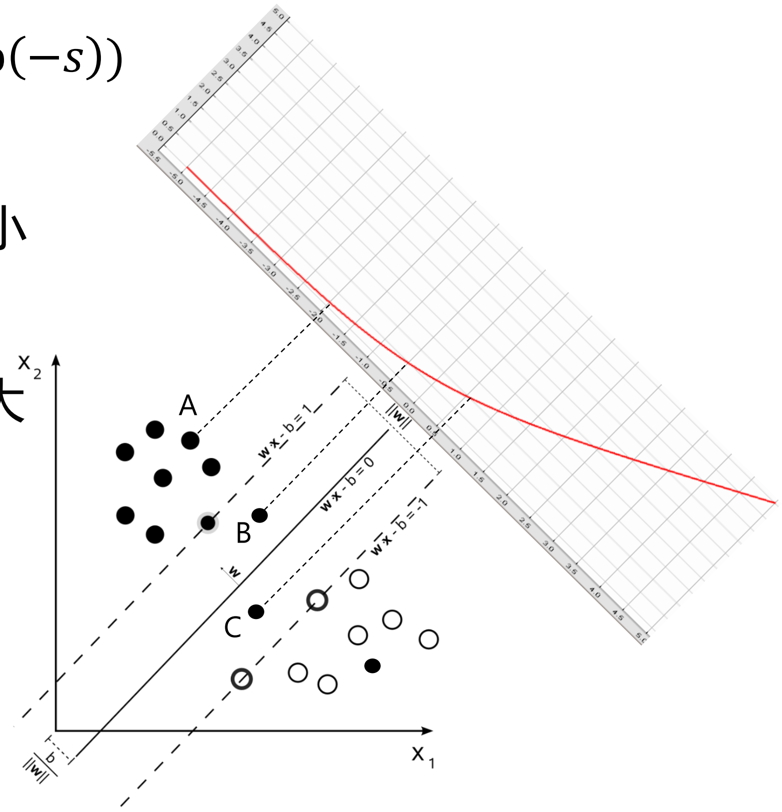
$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

逻辑回归损失是逻辑函数的（负）对数：

$$-\log(1/(1 + \exp(-s)) = \log(1 + \exp(-s))$$

其中$s = w^T x$

- 对于间隔外的正确点（A类点），损失较小
- 对于间隔内的点（B类点），损失较大
- 对于间隔内的错误点（C类点），损失更大



因此，逻辑回归是一种**软间隔分类器**

线性SVM数学推导

- **目标**: **1) 正确分类所有训练数据**

$$wx_i + b \geq 1 \quad if\ y_i = +1$$
$$wx_i + b \leq -1 \quad if\ y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

2) 最大化边距 $M = \frac{2}{|w|}$

等价于最小化 $\frac{1}{2} w^t w$

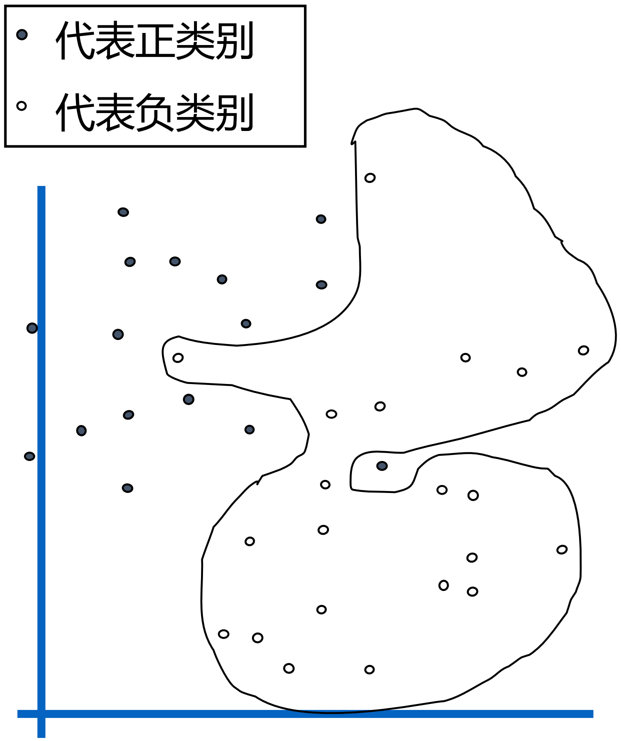
- **我们可以建立一个二次优化问题并求解w和b**

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

SVM支持向量机

含噪声的数据



- 硬边距:到目前为止，我们要求所有数据点正确分类
  - 没有训练误差

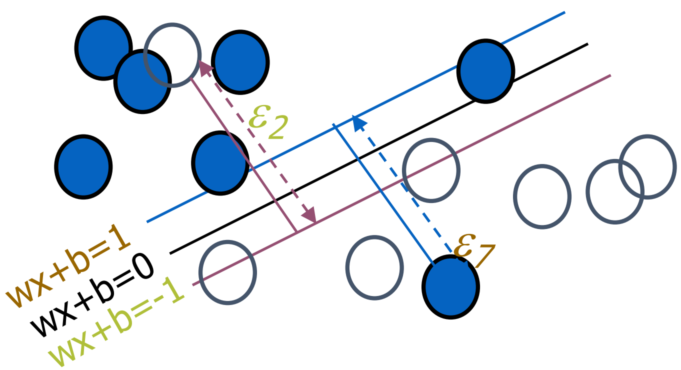
如果训练集包含有噪声呢?
  - 方案1: 使用非常强的核

过拟合!

SVM支持向量机

软边距分类器

添加*松弛变量* ξ_i以允许对困难或带噪声的示例进行错误分类。



我们的二次优化准则应该是什么?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

SVM支持向量机

硬边距 VS 软边距

- **之前的表达式:**

求解$\mathbf{w}$ 和 b 使得

$\Phi(\mathbf{w}) = ½\ \mathbf{w}^T\mathbf{w}$ 最小并对于所有$\{(\mathbf{x_i}, y_i)\}$

$y_i (\mathbf{w}^T\mathbf{x_i} + \mathrm{b}) \geq 1$

- **包含了松弛变量的新的表达式:**

求解$\mathbf{w}$ 和 b 使得

$\Phi(\mathbf{w}) = ½\ \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ 最小并对于所有$\{(\mathbf{x_i}, y_i)\}$

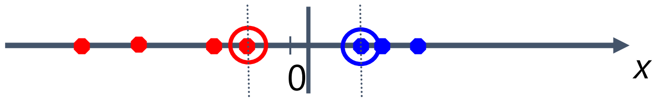
$y_i (\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ 且 $\xi_i \geq 0$ for all i

- **参数 *C* 可以看做是用于避免过拟合的方法.**

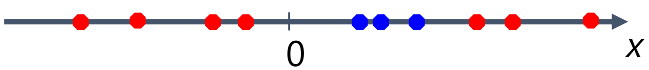
SVM支持向量机

非线性SVM

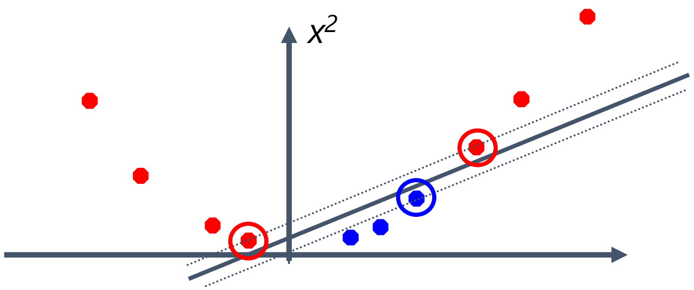
- 对于线性可分的带噪声数据，线性SVM可以工作的很好:



- 但如果数据线性不可分，该怎么处理?



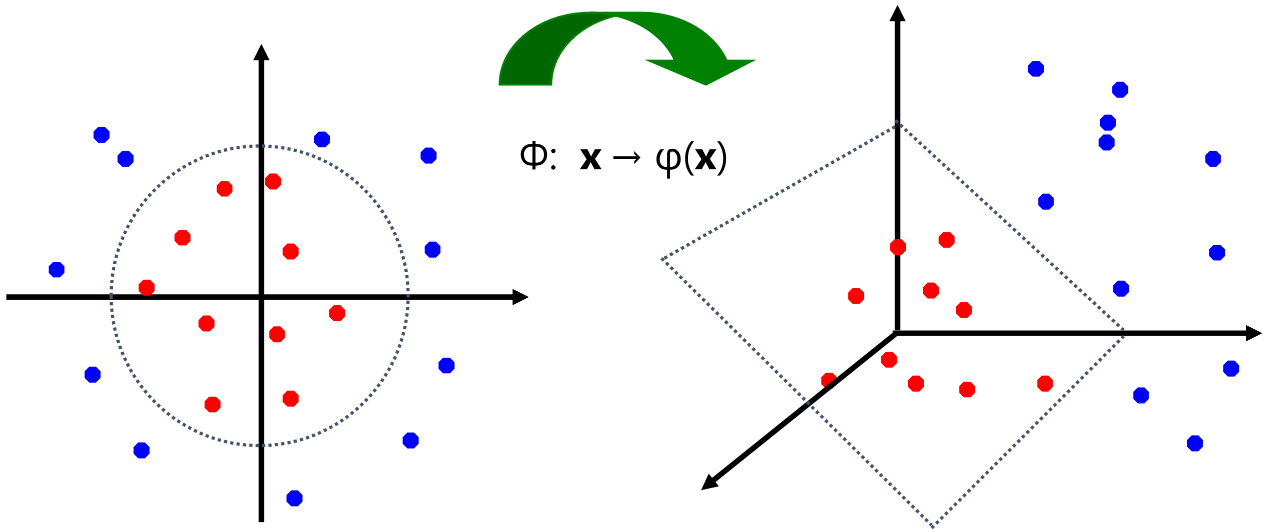
- 将数据映射到更高维的空间:



SVM支持向量机

非线性SVM：特征空间

- 总体思路：原始输入空间始终可以映射到某个训练集可分离的高维特征空间：



$$\phi: \ x = (x_1, x_2) \rightarrow \phi(x) = (x_1^2, x_2^2, x_1 x_2)$$

SVM支持向量机

“核技巧”

- 线性分类器基于向量间的乘积$K(x_i,x_j)=x_i^T x_j$
- 如果所有数据点被映射到一个高纬度空间Φ: x → φ(x), 向量的成绩表示为:

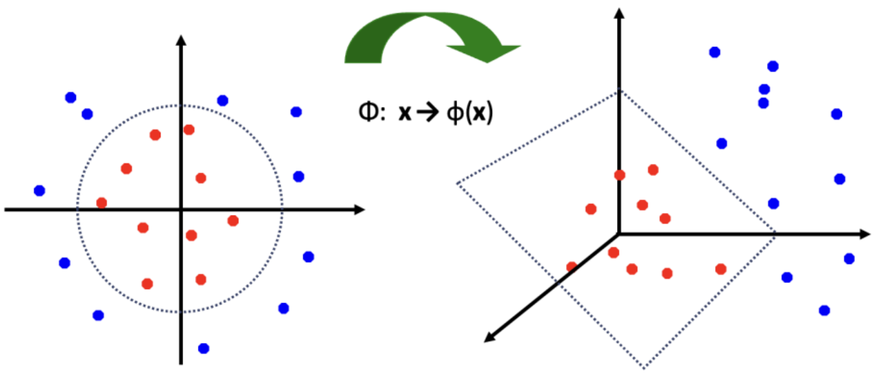
$$K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$$

- 核函数是在扩展的特征空间中与内积相对应的函数.
- 例子:

二维向量$x=[x_1\ \ x_2]$;

$$\phi: \ x = (x_1, x_2) \rightarrow \phi(x) = (x_1^2, x_2^2, x_1x_2)$$

$$\begin{aligned}\langle\phi(x),\phi(z)\rangle &= \big\langle(x_1^2, x_2^2, x_1x_2),(z_1^2, z_2^2, z_1z_2)\big\rangle \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= \langle x_1z_1 + x_2z_2\rangle \\ &= \langle x, z\rangle^2\end{aligned}$$



SVM支持向量机

“核函数的例子”

- 线性核: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$

SVM中核技巧(kernel trick)的作用，一句话概括的话，就是降低计算的复杂度，甚至把不可能的计算变为可能。

- P阶多项式核: $K(\mathbf{x}_i,\mathbf{x}_j) = (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$

- 高斯(径向基函数网络)核:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

- Sigmoid核: $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\beta_0\mathbf{x}_i^T\mathbf{x}_j + \beta_1)$

核函数的优劣

- 劣势：
  - 为给定的问题**选择核函数可能很困难**
  - 对于大型数据集，可能无法存储整个核函数矩阵，可能需要重新计算核函数

- 优势：
  - 核函数在某些特征空间通过点积的方式计算，但**无需知道特征空间以及转换函数**。这就是核函数的有用之处。
  - 使在高维空间中以极低的计算成本寻找线性关系成为可能，这是因为在特征空间中输入图像的内积可以在原始空间中计算出来
  - 不需要数据是真实的向量，可用于字符串、时序数据

非线性SVM回顾

- 支持向量机在特征空间中定位一个分离的超平面，并对该空间中的点进行分类
- 它不需要显式地表示空间，只需定义一个核函数
- 核函数在特征空间中起点积的作用。

SVM的性质

- 调整参数非常耗时
- 适用于小数据集
- 具有处理大型要素空间的能力
  - 复杂性不依赖于特征空间的维数
- 过拟合可以通过软边界方法来控制
- 很好的数学性质：一个简单的凸优化问题，保证收敛到一个单一的全局解

SVM应用

- **SVM 在很多现实中的问题中被成功运用**

-文本（和超文本）分类

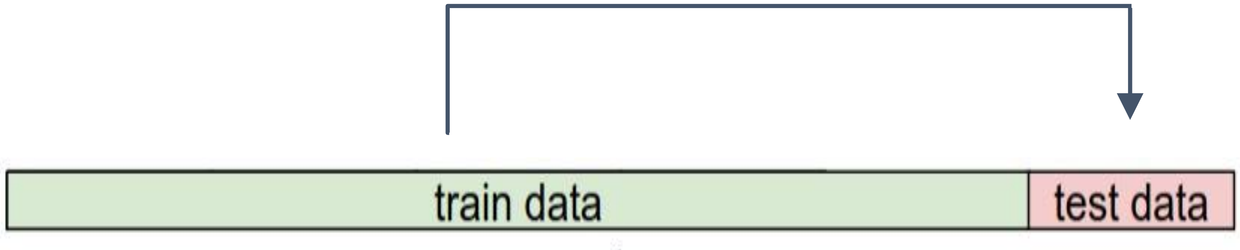
-图像分类

-生物信息学（蛋白质分类，癌症分类）

-手写字符识别

交叉验证

性能评估



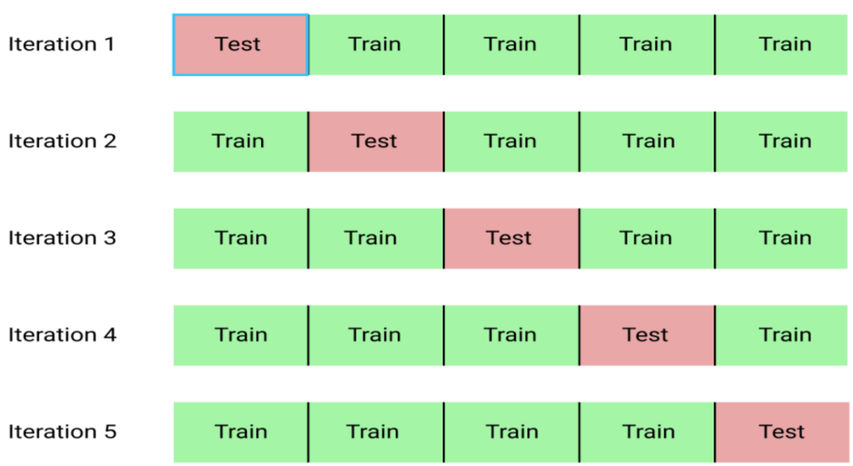
当我们衡量一个分类器的性能时，我们希望使用尽可能多的数据来训练，以便获得最准确的模型。但我们也希望尽可能多地使用数据进行测试，以便获得最准确的测量。

看起来我们只能像上面那样划分数据，我们能做得更好吗？

交叉验证

是的，我们可以像上面那样对数据进行其他的划分。

如果我们保持测试样本的独立性，它们将提供大致独立的测量结果。我们通过使用 k 折交叉验证设计（这里 k=5）来实现这一点：



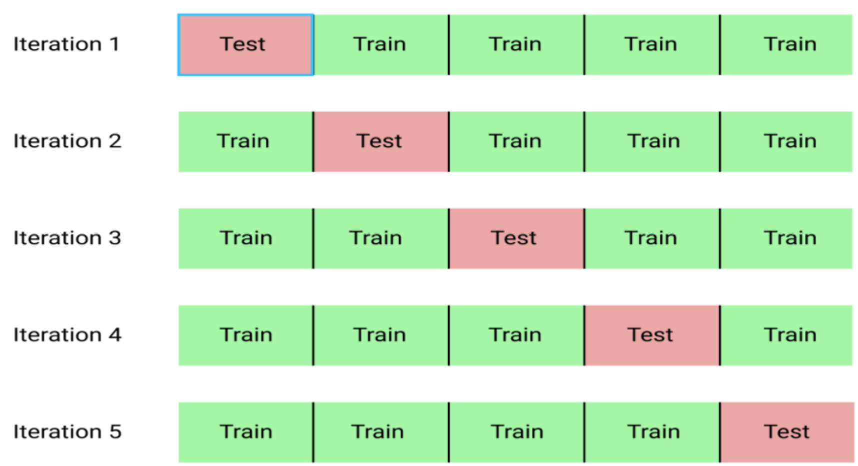
交叉验证

交叉验证

在每次迭代中，我们将4个绿色的折（子集）合并为一个训练集，并在这个训练集上训练模型。然后，我们在该次迭代的红色折（即剩下的那个子集）上评估模型的性能。

最后，我们将5次迭代的性能平均值计算出来。

这样相比于单次运行，可以得到一个更准确的性能估计，尽管标准差（stdev）的改进小于$\sqrt{5}$



总结

总结

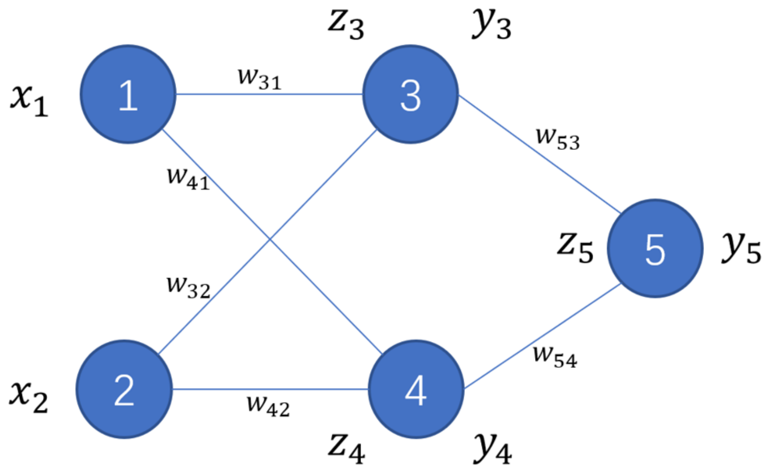
- 我们讨论了正则化来处理过拟合问题
- 我们定义了线性分类器的**间隔**和**合页损失**
- 最大化间隔可以得到支持向量机（SVM）分类器
- 我们还提到了交叉验证作为更准确测试模型的一种方法

致谢

- *Lectures mostly from Dr. John Canny at UC Berkeley*
- *https://www.youtube.com/watch?v=NKuLYRui-NU*
- *https://www.cnblogs.com/XDU-Lakers/p/11770642.html*

考试题型

- 选择题（单选），40分
  - 20道题，每题2分
- 判断题（True or False）20分
  - 20道题，每题1分
- 计算题
  - ER图
  - SQL语句
  - 神经网络及反向传播



考试范围

- Python：字符串命令、数据类型、数组列表
- Numpy：简单函数
- Pandas：dataframe、有空值（NaN）的操作
- 机器学习基础：分类问题、回归问题
- 线性回归、梯度下降
- 过拟合、欠拟合
- 逻辑回归、交叉熵
- 神经网络、反向传播算法、
- 聚类
- 范数、KNN、SVM（核函数）等概念
- 训练集、测试集、验证集