数据挖掘（机器学习）面试--SVM 面试常考问题

应聘数据挖掘工程师或机器学习工程师，面试官经常会考量面试者对 SVM 的理解。

以下是我自己在准备面试过程中，基于个人理解，总结的一些 SVM 面试常考问题（想到会再更新），如有错漏，请批评指正。（大神请忽视）

SVM 的原理是什么？

SVM 是一种二类分类模型。它的基本模型是在特征空间中寻找间隔最大化的分离超平面的线性分类器。（间隔最大是它有别于感知机）

（1）当训练样本线性可分时，通过硬间隔最大化，学习一个线性分类器，即线性可分支持向量机；

（2）当训练数据近似线性可分时，引入松弛变量，通过软间隔最大化，学习一个线性分类器，即线性支持向量机；

（3）当训练数据线性不可分时，通过使用核技巧及软间隔最大化，学习非线性支持向量机。

注：以上各 SVM 的数学推导应该熟悉：**硬间隔最大化（几何间隔）---学习的对偶问题---软间隔最大化（引入松弛变量）---非线性支持向量机（核技巧）。**

SVM 为什么采用间隔最大化？

当训练数据线性可分时，存在无穷个分离超平面可以将两类数据正确分开。

感知机利用误分类最小策略，求得分离超平面，不过此时的解有无穷多个。

线性可分支持向量机**利用间隔最大化求得最优分离超平面**，这时，解是唯一的。另一方面，此时的分隔超平面所产生的分类结果是**最鲁棒**的，对未知实例的**泛化能力最强**。

然后应该借此阐述，几何间隔，函数间隔，及从函数间隔—>求解最小化 1/2 ||w||^2 时的 w 和 b。即线性可分支持向量机学习算法—最大间隔法的由来。

为什么要将求解 SVM 的原始问题转换为其对偶问题？

一、是对偶问题往往更易求解（当我们寻找约束存在时的最优点的时候，约束的存在虽然减小了需要搜寻的范围，但是却使问题变得更加复杂。为了使问题变得易于处理，我们的方法是**把目标函数和约束全部融入一个新的函数，即拉格朗日函数，再通过这个函数来寻找最优点。**）

二、自然引入核函数，进而推广到非线性分类问题。

为什么 SVM 要引入核函数？

当样本在原始空间线性不可分时，可将样本从原始空间映射到一个更高维的特征空间，使得样本在这个特征空间内线性可分。

引入映射后的对偶问题：

$$\begin{aligned} \max_{\lambda} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \phi^T(x_i)\phi(x_j) + \sum_{i=1}^{N}\lambda_i \\ s.t. \quad & \sum_{i=1}^{N}\lambda_i y_i = 0 \\ & 0 \le \lambda_i \le C, \quad i = 1,2,\ldots,N \end{aligned}$$

在学习预测中，只定义核函数 $K(x,y)$，而不是显式的定义映射函数 ϕ。因为**特征空间维数可能很高，甚至可能是无穷维，因此直接计算 $\phi(x)\cdot\phi(y)$是比较困难的。**相反，直接计算 $K(x,y)$比较容易（即直接在原来的低维空间中进行计算，而不需要显式地写出映射后的结果）。

核函数的定义：$K(x,y)=<\phi(x),\phi(y)>$，即在特征空间的内积等于它们在原始样本空间中通过核函数 K 计算的结果。

除了 SVM 之外，任何将计算表示为数据点的内积的方法，都可以使用核方法进行非线性扩展。

svm RBF 核函数的具体公式？

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

Gauss 径向基函数则是局部性强的核函数，其外推能力随着参数 σ 的增大而减弱。

这个核会将原始空间映射为无穷维空间。不过，如果 σ 选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果 σ 选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数 σ，**高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。**

为什么 SVM 对缺失数据敏感？

这里说的缺失数据是指缺失某些特征数据，向量数据不完整。SVM 没有处理缺失值的策略（决策树有）。而 SVM 希望样本在特征空间中线性可分，所以特征空间的好坏对 SVM 的性能很重要。缺失特征数据将影响训练结果的好坏。

SVM 是用的是哪个库？Sklearn/libsvm 中的 SVM 都有什么参数可以调节？

用的是 sklearn 实现的。采用 sklearn.svm.SVC 设置的参数。本身这个函数也是基于 libsvm 实现的（PS: libsvm 中的二次规划问题的解决算法是 SMO）。

SVC 函数的训练时间是随训练样本平方级增长，所以不适合超过 10000 的样本。

对于多分类问题，SVC 采用的是 one-vs-one 投票机制，需要两两类别建立分类器，训练时间可能比较长。

sklearn.svm.SVC(*C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='auto'*, *coef0=0.0*, *shrinking=True*, *probability=False*,*tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape=None*,*random_state=None*)

参数：

l C：C-SVC 的惩罚参数 C?默认值是 1.0

C 越大，相当于惩罚松弛变量，希望松弛变量接近 0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C 值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。

l **kernel** ：核函数，默认是 rbf，可以是‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’

0 – 线性：u'v

1 – 多项式：(gamma*u'*v + coef0)^degree

2 – RBF 函数：exp(-gamma|u-v|^2)

3 –sigmoid：tanh(gamma*u'*v + coef0)

l **degree** ：多项式 poly 函数的维度，默认是 3，选择其他核函数时会被忽略。

l **gamma** ： ‘rbf’,‘poly’ 和‘sigmoid’的核函数参数。默认是’auto’，则会选择 1/n_features

l **coef0** ：核函数的常数项。对于‘poly’和 ‘sigmoid’有用。

l **probability** ：是否采用概率估计？.默认为 False

l **shrinking** ：是否采用 shrinking heuristic 方法，默认为 true

- **tol**：停止训练的误差值大小，默认为 1e-3

- **cache_size**：核函数 cache 缓存大小，默认为 200

- **class_weight**：类别的权重，字典形式传递。设置第几类的参数 C 为 weight*C(C-SVC 中的 C)

- **verbose**：允许冗余输出？

- **max_iter**：最大迭代次数。-1 为无限制。

- **decision_function_shape**：‘ovo’, ‘ovr’ or None, default=None3

- **random_state**：数据洗牌时的种子值，int 值

主要调节的参数有：C、kernel、degree、gamma、coef0。

SVM 如何处理多分类问题？

一般有两种做法：一种是直接法，直接在目标函数上修改，将多个分类面的参数求解合并到一个最优化问题里面。看似简单但是计算量却非常的大。

另外一种做法是间接法：对训练器进行组合。其中比较典型的有**一对一**，和**一对多**。

一对多，就是对每个类都训练出一个分类器，由 svm 是二分类，所以将此而分类器的两类设定为目标类为一类，其余类为另外一类。这样针对 k 个类可以训练出 k 个分类器，当有一个新的样本来的时候，用这 k 个分类器来测试，那个分类器的概率高，那么这个样本就属于哪一类。这种方法效果不太好，bias 比较高。

svm 一对一法（one-vs-one），针对任意两个类训练出一个分类器，如果有 k 类，一共训练出 C(2,k) 个分类器，这样当有一个新的样本要来的时候，用这 C(2,k) 个分类器来测试，每当被判定属于某一类的时候，该类就加一，最后票数最多的类别被认定为该样本的类。

参考资料：

李航 《统计学习方法》（强烈推荐）

周志华 《机器学习》

部分资料整理于网上。

支持向量机: Kernel http://blog.pluskid.org/?p=685&cpage=1