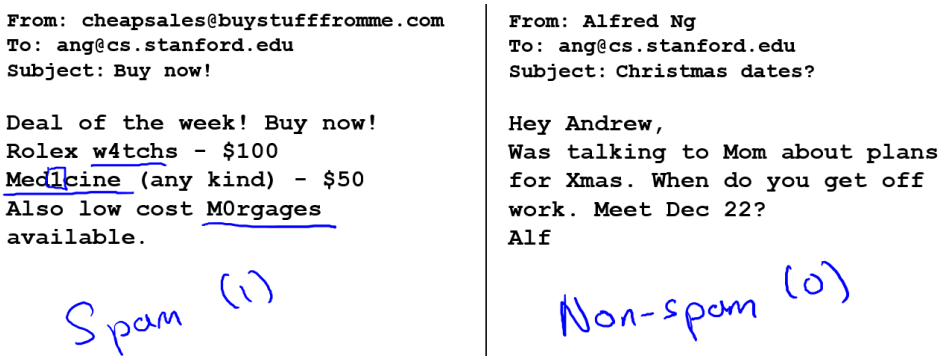
11 机器学习系统设计

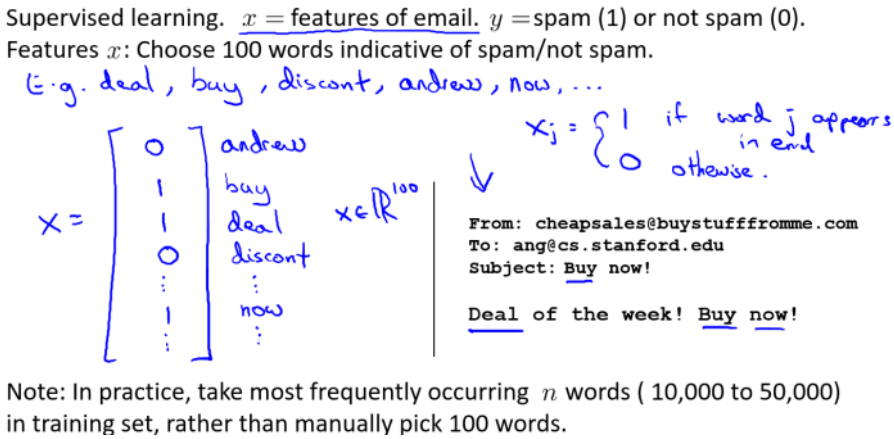
2022年11月11日 21:09

11.1确定执行的优先级

建立一个垃圾邮件分类器



垃圾邮件里有拼错的单词。用1和0分别表示垃圾邮件和非垃圾邮件。



比如用逻辑回归就可以训练一个分类器。提出一个含有100个单词的列表来区分垃圾邮件和非垃圾邮件。检查定义的单词是否在邮件中出现，并用特征向量表示。实际的做法是挑选出出现频率最多的n个单词，一般n在10000到50000之间，然后将其作为特征向量。

当我们在做这个分类器的时候，会遇到的一个问题就是如何在有限的时间限制条件下让分类器具有高精准度和低错误率。

- Collect lots of data
  - E.g. “honeypot” project.
- Develop sophisticated features based on email routing information (from email header).
- Develop sophisticated features for message body, e.g. should “discount” and “discounts” be treated as the same word? How about “deal” and “Dealer”? Features about punctuation?
- Develop sophisticated algorithm to detect misspellings (e.g. m0rtgage, med1cine, w4tches.)

- 一种可以想到的方法是收集大量的数据
- 用更复杂的特征变量来描述，比如从邮件标题构建
- 关注邮件的主体部分(正文)，并构建更加复杂的特征。
- 设计一些更复杂的算法来检测出单词中故意出现的拼写错误

11.2误差分析

如果要开发机器学习的应用

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- Error analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

- 一般先通过一个简单的算法快速实现，然后通过交叉验证来测试数据。
- 然后画出相应的学习曲线，检验误差来确定模型是否存在高偏差或高方差问题，然后决定是否使用更多的数据或特征。
- 还有一件很重要的事情就是误差分析。关注交叉验证集的错误情况。以垃圾邮件的分类问题为例，可以寻找被错误分类的邮件有什么共同规律，这可以启发我们设计新的特征。

一个具体的例子：

m_{CV} = 500 examples in cross validation set
Algorithm misclassifies 100 emails.
Manually examine the 100 errors, and categorize them based on:
(i) What type of email it is — pharma, replica, steal passwords, ...
(ii) What cues (features) you think would have helped the algorithm classify them correctly.

对于一个垃圾邮件分类器，在交叉验证集中有500个样例，而测试的时候分类器错误地分类了100个。手动查看这100个错误然后为它们分类。通过查看邮件的类型

Pharma: 12
Replica/fake: 4
Steal passwords: 53
Other: 31

Deliberate misspellings: 5
(m0rgage, med1cine, etc.)
Unusual email routing: 16
Unusual (spamming) punctuation: 32

从而针对于相应类型的邮件看是否有更好的特征来帮助正确分类。

Should discount/discounts/discounted/discounting be treated as the same word?
Can use "stemming" software (E.g. "Porter stemmer")
universe/university.

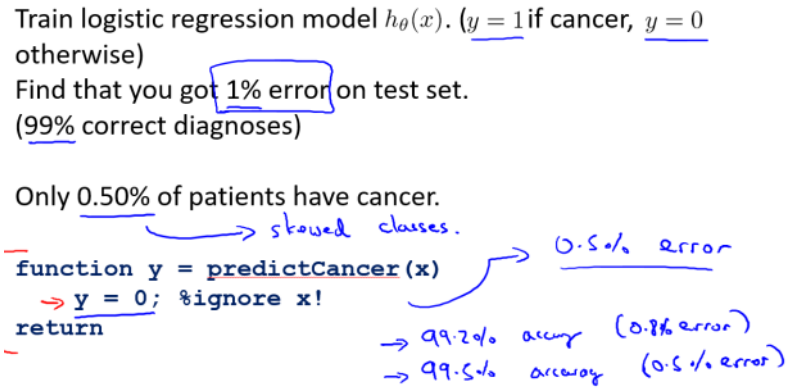
最后，在改进算法时，另一个技巧是对算法有一个数值估计的方法。比如几个相同词源的单词是否应该被当作一个单词，那么很难抉择是否应该这样做，即使通过误差分析也很难确定。而为了快速判断词干提取软件是否对算法有益，可以尝试一下看它是否有效果。通过数值方法来评估算法效果将会非常有用。

Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.
Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.
Without stemming: 5% error With stemming: 3% error
Distinguish upper vs. lower case (Mom/mom): 3.2%

最常想到的就是计算交叉验证集的错误率来判断。

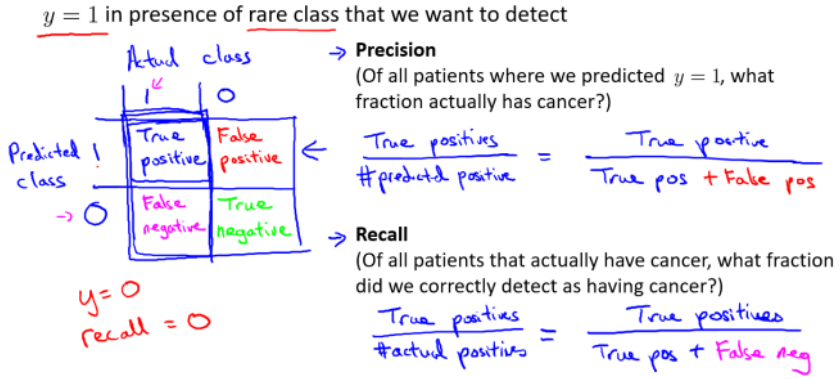
11.3不对称分类的误差评估

以之前的癌症分类为例，



通过构建逻辑回归模型对肿瘤是恶性还是良性进行判断。但是假设一种情况，那就是在测试集中只有0.5%的人是真正的的癌症患者，那么对于一个正确率为99%的模型，它的效果就显得不是那么好。比如说，我们在一个预测算法中忽略特征量x，对于任意的输入值都把它判断为患了癌症，那么这个模型的准确率甚至可以达到惊人的99.5%。这种情况发生在正例和反例的比例达到一个非常极端的时候，也把0.5%的这一类例子叫做偏斜类(skewed class)。所以这是只使用错误率来对模型评估时很可能发生的问题：即对于具有偏斜类的分类问题，虽然我们能得到很高的正确率和很低的错误率，但是我们并不知道分类质量是否真的提高了。

查准率或召回率是一个很好的衡量指标。



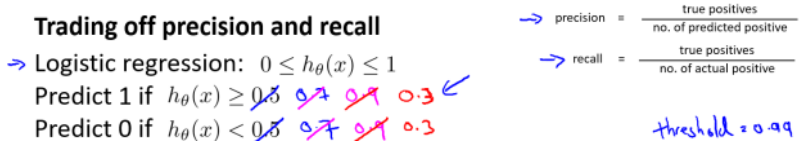
混淆矩阵：

	实际正例	实际反例
预测正例	真正例(True positive)	假正例(False positive)
预测反例	假反例(False negative)	真反例(True negative)

- **查准率**(Precision):对于我们预测的所有患有癌症的人，有多大比率的人是真正患有癌症的。越高越好。
  $\frac{真正例}{预测正例}$
- **查全率**(Recall):对于实际患有癌症的人，有多大比率正确预测了他们患有癌症。越高越好。
  $\frac{真正例}{实际正例}$

一般将偏斜类(上例中比例为0.5%的那一类)作为正例。

11.4查准率和查全率的权衡



假设病人患癌症那么y=1，否则为0。我们用逻辑回归模型训练了数据，当假设函数的

值大于0.5时就预测为正例，当假设函数的值小于0.5时就预测为反例。针对这个例子，我们假设更希望在非常确信的状态下告诉病人他患了癌症，

→ Suppose we want to predict $y = 1$ (cancer) only if very confident.

→ Higher precision, lower recall.

那么我们可以修改算法，不再将临界值设置为0.5，而令它大于0.7。此时模型会有比较高的查准率，但查全率相应地会降低。

从另一个角度，假设我们希望尽量避免漏掉患有癌症的人，即我们希望尽力避免假反例。也就是说，一个病人本来患了癌症，但是我们告诉他没有患癌症的后果非常严重，

→ Suppose we want to avoid missing too many cases of cancer (avoid false negatives).

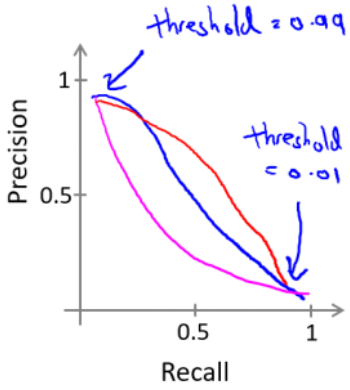
→ Higher recall, lower precision.

那么我们可以不设置那么高的临界值，比如0.3。此时模型具有较高的查全率，但是查准率会比较低。

More generally: Predict 1 if $h_\theta(x) \geq$ threshold. ←

所以算法的倾向一般取决于我们想要更高的查准率还是查全率。

查准率和查全率的曲线大致如下：



那么我们有没有办法自动选取一个合适的临界值？

How to compare precision/recall numbers?

	Precision(P)	Recall (R)
→ Algorithm 1	0.5	0.4
→ Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

Predict y=1 all the time

Average: ~~$\frac{P+R}{2}$~~

F_1 Score: $2\frac{PR}{P+R}$

P=0 or R=0 ⇒ F-score = 0.

P=1 and R=1 ⇒ F-score = 1

一种我们可能会想到的方法就是求查全率和查准率的平均值来看哪个模型有较高的平均值，但它并不是一个很好的方法。如上图所示的算法3，它的查全率很高、查准率很低，但它的平均值很大，显然这个算法并不好。

因此，F值公式是一个好的评价指标。

$F = 2\frac{PR}{P+R}$

它结合了两个指标。

11.5机器学习数据

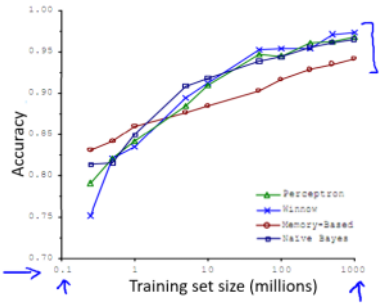
E.g. Classify between confusable words.

{to, two, too} {then, than}

For breakfast I ate two eggs.

Algorithms

- Perceptron (Logistic regression)
- Winnow
- Memory-based
- Naïve Bayes



有研究表明，很多算法在外部空间上的性状相似，而增加训练样本能够改善其性能，即使它可能是一个比较劣质的算法。

机器学习中有一句经典的话：

“It’s not who has the best algorithm that wins.

It’s who has the most data.”

不是拥有最好算法的人能成功，而是拥有最多数据的人能成功。

Large data rationale

Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.

Example: For breakfast I ate two eggs.

Counterexample: Predict housing price from only size (feet^2) and no other features.

Useful test: Given the input x, can a human expert confidently predict y?

当特征量和所提供的信息足够多时，相关领域的专家可以预测出来；而当所具备的特征和信息不足时，即使是相关的专家也不一定能预测。那么算法也是这样吗？

Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units). low bias algorithms.

→ $J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) low variance

→ $J_{train}(\theta) \approx J_{test}(\theta)$

$J_{test}(\theta)$ will be small

对于一个具有很多参数的学习算法，当训练数据很多，远远大于参数数量时，此时训练误差很小，且不太会发生过拟合，那么此时训练误差和测试误差都很小且十分接近。由于参数够多且不会过拟合，可以认为这个模型是低方差、低偏差的。