Stanford CS229 Machine Learning, Kernels, 2022, Lecture 7

Link on YouTube: Stanford CS229 Machine Learning, Kernels, 2022, Lecture 7

Feature Function

Introduction: Cubic Polynomial Regression, where data set is $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\}$

在基本线性回归(Linear Regression)模型$h_\theta(x) = \theta x + \theta_0$中[a]，我们仅仅使用了$x$ 自身进行数据拟合，但我们常需要更复杂的模型，例如三次多项式(cubic polynomial) 模型：

$$h_\theta(x) = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x_1 + \theta_0,\ x \in \mathbb{R}$$

显然模型$h_\theta(x)$关于x是非线性的，但关于参数θ却是线性的[b]。我们只需做出一些改变，就可以将关于x的非线性模型变为关于θ的线性模型。

[a]其中θ代表全体参数

[b]我们关注的核心实际上是在参数空间中对θ进行优化

Feature Map / Feature Extractor (cubic polynomial regression case)

定义函数$\phi : \mathbb{R} \to \mathbb{R}^4$， $\phi(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix}^T$，这样模型就可以写为

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 \end{bmatrix} \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix}^T = \theta^T \phi(x)$$

如此一来我们原本的模型输入从$\begin{bmatrix} 1 & x \end{bmatrix}$(二维)变成$\phi(x)$(四维)，因此我们构建了新数据集：

$$\{(\phi(x^{(1)}, y^{(1)}), \cdots (\phi(x^n), y^{(n)})\}$$

其中函数$\phi(\cdot)$被称为特征提取函数(feature function / feature extractor)，$\phi(x)$被称为特征(features)，x 被称为属性(attribute)。如果$x \in \mathbb{R}^d$，那么相应的$\theta \in \mathbb{R}^d$；在经过$\phi : \mathbb{R}^d \to \mathbb{R}^p$的作用后，相应的$\theta \in \mathbb{R}^p$。

Kernel Trick / Kernelized

Basic Settings

在新数据集上做线性回归，并使用梯度下降(GD)进行优化，学习率为α，此时损失函数和参数θ的更新过程分别为式(1a)(1b)

$$loss = \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi \left(x^{(i)} \right) \right)^2 \tag{1a}$$

$$\theta := \theta + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)}) \tag{1b}$$

考虑$x \in \mathbb{R}^d$，使用三次多项式回归，那么$\phi(x) \in \mathbb{R}^p$，其中$p = 1 + |\{x_i \cdot x_j\}| + |\{x_i \cdot x_j \cdot x_k\}| = 1 + d + d^2 + d^3$(考虑重复情形)。因此每一次参数$\theta$的更新都会有$O(np)$的计算量，这是十分巨大的。

Kernel Trick

Proposition 1 (Rey Observation)**.** 若$\theta^0=0$, 那么θ 可以表示为*features* 的线性组合，即

$$\theta=\sum_{i=1}^{n}\beta_i\phi(x^{(i)}) \tag{2}$$

其中$\beta_1,\cdots\beta_n\in\mathbb{R}$，$\theta^0$ 是θ 的初始值。

Proof. 我们使用数学归纳法(induction)。
当iteration = 0 时，$\theta=0=\sum_{i=1}^{n}0\cdot\phi\left(x^{(i)}\right)$显然是features 的线性组合；
当iteration = 1 时，$\theta:=\theta+\alpha\sum_{i=1}^{n}(y^{(i)}-\theta^T\phi(x^{(i)}))\phi(x^{(i)})=\alpha\sum_{i=1}^{n}y^{(i)}\phi(x^{(i)})$，此时$\beta_i:=\alpha y^{(i)}$
假设iteration = t 时结论成立，即$\theta=\sum_{i=1}^{n}\beta_i\phi\left(x^{(i)}\right)$，则iteration = $t+1$ 时

$$\begin{aligned}&\theta:=\theta+\alpha\sum_{i=1}^{n}\left(y^{(i)}-\theta^T\phi(x^{(i)})\right)\phi(x^{(i)})=\sum_{i=1}^{n}\left(\left(\beta_i+\alpha\left(y^{(i)}-\theta^T\phi\left(x^{(i)}\right)\right)\right)\phi\left(x^{(i)}\right)\right)\\&\beta_i:=\beta_i+\alpha\left(y^{(i)}-\theta^T\phi\left(x^{(i)}\right)\right)\end{aligned} \tag{3}$$

□

Trick 1: parameters storage 已知$\theta\in\mathbb{R}^p$，但是现在我们不需要直接存储θ 而是存储系数β_i，这只需要n 个存储，不妨假设$p>>n$，那么这将带来一些计算量的降低。

Problem 1: still related to θ 从式(3)可以看到β的更新实际上与θ相关，仍需$O(p)$的计算量，因此我们需要找到β只依赖于自身的更新模式：

$$\begin{aligned}\beta_i&:=\beta_i+\alpha\left(y^{(i)}-\theta^T\phi\left(x^{(i)}\right)\right)=\beta_i+\alpha\left(y^{(i)}-\left(\sum_{j=1}^{n}\beta_j\phi\left(x^{(j)}\right)\right)^T\phi\left(x^{(i)}\right)\right)\\&=\beta_i+\alpha(y^{(i)}-\sum_{j=1}^{n}\beta_j\langle\phi(x^{(j)}),\phi(x^{(j)})\rangle)\end{aligned} \tag{4}$$

Problem 2: inner product costs a lot 但是可以看到在计算内积$\langle\phi(x^{(j)}),\phi(x^{(j)})\rangle$时，仍然需要$O(np)$的计算量，下面有两个简化方式(**Trick 2**):
① **preprocessed:** $\langle\phi(x^{(j)}),\phi(x^{(j)})\rangle$ 可以被预处理(preprocessed)并存储，因为$\forall i,j,$，内积计算过一次后就无需重复计算了
② **formal compute:** 无需精确地将复杂的$\phi(\cdot)$带入$\langle\phi(x^{(j)}),\phi(x^{(j)})\rangle$中也可以进行计算：

$$\begin{aligned}&\langle\phi(x),\phi(z)\rangle=[1,x_1,\cdots,x_d,x_1^2,\cdots,x_d^2,x_1^3,\cdots,x_d^3][1,x_1,\cdots,x_d,x_1^2,\cdots,x_d^2,x_1^3,\cdots,x_d^3]^T\\&=1+\sum_{i=1}^{d}x_iz_i+\sum_{i=1,j=1}^{d}x_ix_j\cdot z_iz_j+\sum_{i,j,k=1}^{d}x_ix_jx_kz_iz_jz_k\\&=1+\langle x,z\rangle+\sum_{i=1}^{d}x_iz_i\sum_{j=1}^{d}x_jz_j+\sum_{i=1}^{d}x_iz_i\cdot\sum_{j=1}^{d}x_jz_j\cdot\sum_{k=1}^{d}x_kz_k=1+\langle x,z\rangle+\langle x,z\rangle^2+\langle x,z\rangle^3\end{aligned}$$

Algorithm and Time complexity

可以看到计算单个内积的时间复杂度为$O(d)$。称$K:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$, $K(x,z)=\langle\phi(x),\phi(z)\rangle$为核函数(kernel function)。这样的算法称为Kernel method for regession，最终的算法为：

Algorithm 1 Algorithm for Kernel Computation and β Update

Input: Data points $\{x^{(i)}\}_{i=1}^n$, labels $\{y^{(i)}\}_{i=1}^n$, learning rate α
Initialize: $\beta=0\in\mathbb{R}^n$
for $i=1,\ldots,n$ **do**
  for $j=1,\ldots,n$ **do**
    Compute the kernel $K(x^{(i)},x^{(j)})=\langle\phi(x^{(i)}),\phi(x^{(j)})\rangle$
  end for
end for
for $i=1,\ldots,n$ **do**
  $\beta_i\leftarrow\beta_i+\alpha\left(y^{(i)}-\sum_{j=1}^n\beta_jK(x^{(i)},x^{(j)})\right)$
end for

考虑维度i,j后，计算所有的kernel 的总时间复杂度为$O(n^2d)=O(n^2)$，因此对于β，每一次迭代的时间复杂度为$O(n^2)$。可以看到单次迭代β的时间复杂度变化为$O(np)\to O(n^2)$，因此若$n<<p$，那么就会带来很大的计算量提升。[a]

[a]整个算法的时间复杂度为$O(n^2d)+O(n^2\cdot T)$，其中T是迭代次数

Design Kernel Function

Change the algorithm flow

从上述推导过程来看，我们将$K(\cdot,\cdot)$从特征函数的内积形式化成为原数据的内积进而减少了计算量。在进行测试时，给定一个x，根据式(3)和Proposition 1需要计算

$$\theta^T\phi(x)=\left(\sum_{i=1}^n\beta_i\phi\left(x^{(i)}\right)\right)^T\phi(x)=\sum_{i=1}^n\beta_iK\left(x^{(i)},x\right)$$

[a]，因此我们的算法最终仅与核函数K有关，而核函数是特征函数的内积形式，因此显示定义了特征函数就显式定义了核函数，反过来直接定义核函数也就隐式定义了特征函数，因此[b]：

Design a good $\phi\Leftrightarrow$ Design a good K

Theorem 1 (necessary and equvalant condition for designing a K)**.** *Suppose $x^{(1)},\cdots,x^{(n)}$ are n data points, and let $K\in\mathbb{R}^{n\times n}$ be kernel matrix, in which $K_{ij}=K(x^{(i)},x^{(j)})$. K is a valid kernel function $\Leftrightarrow\forall\,x_1^{(n)}\cdots,x^{(n)}$, the kernel function K is PSD (positive semi-definite).*

Proof. 暂略 □

如此一来我们的工作流程就反过来变为了：
设计一个好的kernel function $K\to$ 确定K 有效(通过Thm 1 或者解出$\phi(\cdot)$) $\to$ 运行算法

[a]可以看到此时测试时仍然需要训练数据，与之前训练好之后就无需训练数据只需训练好的参数的情况不同

[b]这里的K的设计要make sense，要有意义

Extended content

常用的核函数有

1. $K(x,z)=(\langle x,z\rangle+c)^k$, for $k=2$, $\phi\left(x\right)=\begin{bmatrix} c \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ x_1^2 \\ \vdots \\ x_d^2 \end{bmatrix}$

2. Gaussian Kernel: $K(x,z)=\exp(-\frac{||x-z||_2^2}{2\sigma^2})=\langle\phi(x),\phi(z)\rangle$，但是实际上$\phi(\cdot)$很复杂，是一个无穷维形式

有时特征也被视为相似性度量(similarity metric)，可以将$K(x,z)$看作一种测量x和z的方式(measure)

kernel method 真正的改变是将$O(np)\to O(n^2)$，当$n << p$时这较有效，但是现在的数据集往往很大，因此kernel method 现在并不是一个很高效和常用的方法；同时核函数的设计并不很容易，同时可解释性也是一个困难。事实上*NN*也可以写为$\theta^T\phi_w(x)$，其中w代表*NN*中的参数，而此ϕ_w是直接从数据中学习得到的，而非人工设计的，但是有效性却出奇的好。