Python Homework

郑志浩 (11521075)

June 1, 2016

1 Homework1

Goal:Implement polynomial curve fitting in python.

1.1 思路或原理

多项式为下式

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1}$$

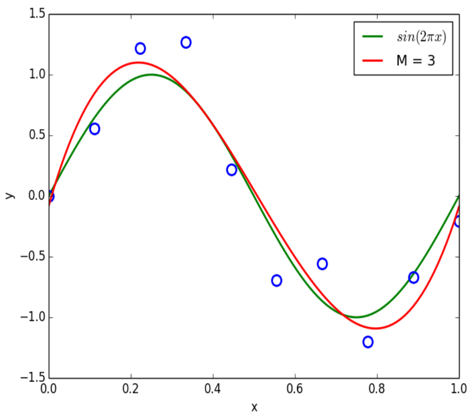
来拟合离散的点。拟合程度用二次函数来表示,等式右边的第二项是正则项，用于惩罚大系数值

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2 \tag{2}$$

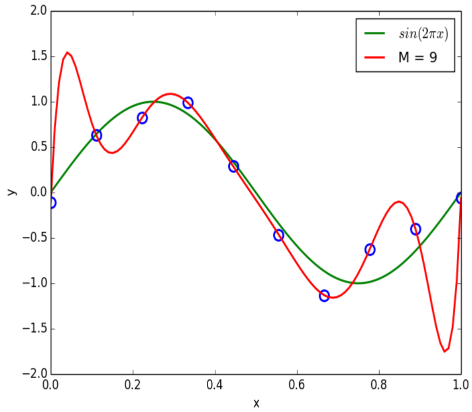
上述多项式化为矩阵形式为$\mathbf{y} = \mathbf{P}(\mathbf{x}) \cdot \mathbf{w}$,要求的目标为$\arg\min_w \|y - P(x) \cdot w\|^2$,最终得到$\mathbf{w} = (\mathbf{P}^{\mathrm{T}}\mathbf{P} + \lambda \mathbf{I})^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{y}$，其中$\mathbf{p}(x) = (x^0, x^1, \cdots, x^M)^T$,$\mathbf{w}= (w_0, w_1, \cdots, w_M)^T$，$\mathbf{y}= (y_0, y_1, \cdots, y_M)^T$，$y_j = \mathbf{p}(\mathbf{x_j})\mathbf{w}$。

1.2 实验结果

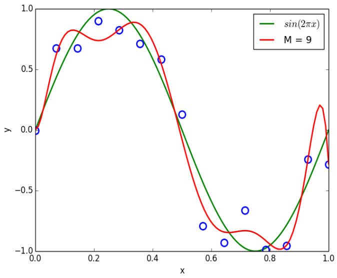
1.sample the function curve of y=sin(x) with Gaussian noise

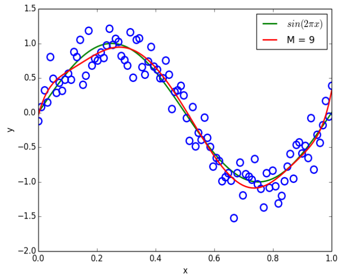


2.fit degree 3 and 9 curves in 10 samples

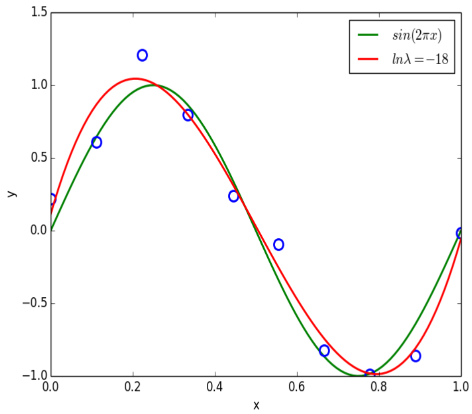


3.fit degree 9 curves in 15 and 100 samples





4.fit degree 9 curve in 10 samples but with regularization term



2 Homework2

Goal:Represent digits ’3’ in 2D
(1)convert data from the UCI
(2)perform PCA over all digit ’3’ with 2 components
(3)plot the PCA results as below

2.1 思路或原理

PCA的作用是用降维的数据来表示原始数据，同时尽可能地保留原始数据样貌，若将数据降为到p维的话，数据点y可以看作特征空间基的线性组合：

$$y \approx \widetilde{x} + \sum_{i=1}^{p} w_i x_i, y, x_i \in \mathbb{R}^{d\times 1}$$

所以将PCA形式化后就为

$$\arg\min_{X,W} \|Y - X^T W\|, Y \in \mathbb{R}^{d\times N}, X \in \mathbb{R}^{p\times d}, W \in \mathbb{R}^{p\times N}$$

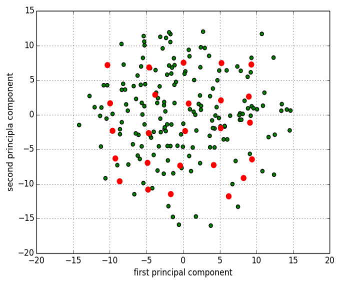
于是采用特征分解的方法来求解上述问题，证明细节不给出，下面给出算法过程。
输入：样本集$D = \{x_1, x_2, \cdots, x_m\}$
过程：

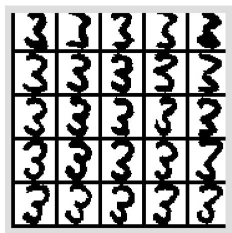
1.对所有样本进行中心化：$x_i \leftarrow x_i - \frac{1}{m}\sum_{i=1}^{m} x_i$；
2.计算样本协方差矩阵XX^T；
3.协方差矩阵XX^T做特征分解；
4.取最大的d'个特征值所对应的特征向量$w_1, w_2, \cdots, w_{d'}$

输出：投影矩阵$W = (w_1, w_2, \cdots, w_{d'})$。

2.2 实验结果

取199个数据点最小包围框，六等分每个边界，取最靠近内部交织出的网格点附近的点作为显示的数据点，具体点的位置用红点标出，实验的输出如下图示：





3 Homework3

Goal:implement MOG in 2D case
1.Generate 2D Gaussian distribution
2.E-M method.

3.1 思路或原理

1.利用$numpy.random.multivariate_normal(mean, cove, size)$来生成size维度的高斯分布
2.利用E-step和M-step两步迭代直到收敛为止

其中E-step为：

$$Q^i(z^{(i)}) = p(z^{(i)}|x^{(i)};\theta) \tag{3}$$

M-step为：

$$\theta := \arg\max_{\theta} \sum_{i=1}^{m} \sum_{z^{(i)}} Q^i(z^{(i)}) \log \frac{p(x^{(i)})}{Q^i(z^{(i)})} \tag{4}$$

3.2 代码说明

通过改变main里面的mean1,cov1,mean2和cov2等来调整高斯分布，最后通过将组合等数据投入到EM函数里就能获得结果

3.3 实验结果

通过EM算法迭代得到的试验结果如下：

(e) 输入未分类数据 (f) 得到分类数据

图 1: EM算法实验结果。

4 Homework4

Goal:Implement the Levenberg-Marquardt method

4.1 思路或原理

利用符号记录函数，然后通过下述算法求解最优解：

begin

$k := 0$; $\nu := 2$; $\mathbf{x} := \mathbf{x}_0$

$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x})$; $\mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

$found := (\|\mathbf{g}\|_\infty \le \varepsilon_1)$; $\mu := \tau * \max\{a_{ii}\}$

while (**not** $found$) **and** $(k < k_{\max})$

$k := k+1$; Solve $(\mathbf{A} + \mu\mathbf{I})\mathbf{h}_{\mathrm{lm}} = -\mathbf{g}$

if $\|\mathbf{h}_{\mathrm{lm}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$

$found :=$ **true**

else

$\mathbf{x}_{\mathrm{new}} := \mathbf{x} + \mathbf{h}_{\mathrm{lm}}$

$\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\mathrm{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\mathrm{lm}}))$

if $\varrho > 0$ {step acceptable}

$\mathbf{x} := \mathbf{x}_{\mathrm{new}}$

$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x})$; $\mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

$found := (\|\mathbf{g}\|_\infty \le \varepsilon_1)$

$\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}$; $\nu := 2$

else

$\mu := \mu * \nu$; $\nu := 2 * \nu$

end

其中$\mathrm{J}(\mathbf{x})_{i,j} = \frac{\partial \mathrm{f}_i}{\partial \mathrm{x}_j}$, $\varrho = \frac{F(x) - F(x + h_{lm})}{L(0) - L(h_{lm})}$,以及

$$
\begin{aligned}
L(0) - L(h_{lm}) &= -\mathbf{h}_{\mathrm{lm}}^{\mathrm{T}}\mathrm{J}^{\mathrm{T}}\mathrm{f} - \frac{1}{2}\mathbf{h}_{\mathrm{lm}}^{\mathrm{T}}\mathrm{J}^{\mathrm{T}}\mathrm{J}\mathbf{h}_{\mathrm{lm}} \\
&= -\frac{1}{2}\mathbf{h}_{\mathrm{lm}}^{\mathrm{T}}(2\mathrm{g} + (\mathrm{J}^{\mathrm{T}}\mathrm{J} + \mu\mathbf{I} - \mu\mathbf{I})\mathbf{h}_{\mathrm{lm}}) \\
&= \frac{1}{2}\mathbf{h}_{\mathrm{lm}}^{\mathrm{T}}(\mu\mathbf{h}_{\mathrm{lm}} - \mathrm{g})
\end{aligned}
$$

4.2 实验结果

在LMA.py中提供了两个例子，其中例子2就是著名的Powell's问题，迭代次数达到了最大次数100次，得到的值为$[-3.30676797\times 10^{-07}, -4.06964635\times 10^{-03}]^T$。例子1较简单就不详述，只需要25次就迭代收敛了，当然也可以自己再再LMA.py文件中写更多的测试函数。

5 Homework5

Goal:Implement (simplified) SVM method
1.input 2D data and their label (in two classes)
2.implement quadratic programming
3.output (and plot) classification results

5.1 思路或原理

SVM问题形式化：
原问题：

$$\min_{w,b}\frac{1}{2}\|w\|^2$$

$$s.t.\quad y_i(w^ix_i+b)\geqslant 1, i=1,2,\cdots,m$$

对偶问题：

$$\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jy_iy_j\langle x_i,x_j\rangle$$

$$s.t.\quad \sum_{i=1}^{m}\alpha_iy_i=0$$

$$s.t.\quad \alpha_i\leqslant 0, i=1,2,\cdots,m$$

求解后即可得到

$$f(x)=\sum_{i=1}^{m}\alpha_iy_i\langle x,x_i\rangle+b$$

其中$\langle x_i,x_j\rangle$若是简化的SVM即为线性核$x_i^Tx_j$，若需要将他们投到高纬空间需要一些核函数，例如高斯核函数。
拉格朗日乘子法求解等书二次规划问题

$$\min f(x)=\frac{1}{2}x^TGx+r^Tx$$

$$Ax=b$$

定理 当上述问题中的矩阵G是半正定（正定）矩阵时，局部解x^*是全局最优解，这时λ^*为相应的乘子的充分必要条件是：x^*，λ^*是线性方程组

$$\begin{pmatrix} G & A^T \\ A & O \end{pmatrix}\begin{pmatrix} x \\ \lambda \end{pmatrix}=\begin{pmatrix} -r \\ b \end{pmatrix}$$

的解。 有效集法求解凸二次规划问题(ESM)：

⑴取初始可行点x^1（一般都是比较难获取，在SVM的环境下，比较好取得），即x^1满足

$$\mathbf{\alpha_i^Tx^1}-b_i=\mathbf{0}, i\in\mathbf{E}, \mathbf{\alpha_i^Tx^1}-b_i\leqslant\mathbf{0}, i\in\mathbf{I},$$

确定x^1处的有效集约束指标集

$$I(x^1)=\{i|\alpha_i^Tx^1-b_i=0, i\in I\},$$

置$k=1$。

⑵通过拉格朗日乘子法求解等式二次规划问题：

$$\min\frac{1}{2}d^TGd+\nabla f(x^k)^Td;$$

$$s.t.\quad \alpha_i^Td=0, i\in E\bigcup I(x^k),$$

得到d^k。 (3)若$d^k = 0$,则计算相应的拉格朗日乘子λ^k（λ^k为上述拉格朗日乘子法求的的相应乘子）。若$\lambda_i^k \geqslant 0, \forall i \in I(x^k)$，则停止计算；否则求

$$\lambda_q^k = \min \lambda_i^k | i \in I(x^k),$$

并置$x^k + 1 = x^k, I(x^k + 1) = I(x^k) - \{q\}, k := k + 1$，转步骤（2）。

(4)若$\mathbf{d^k} \neq \mathbf{0}$,则计算

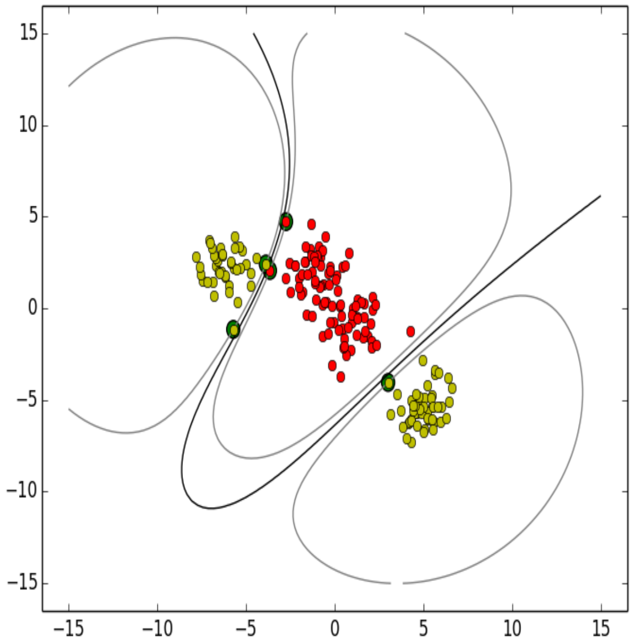
$$\hat{\alpha}_k = \min\{\frac{b_i - \alpha_i^T x^k}{\alpha_i^T d^k} | \alpha_i^T d^k > 0, i \in I(x^k)\}$$

$$= \frac{b_p - \alpha_p^T x^k}{\alpha_p^T d^k}.$$

取$\alpha_k = \min\{\hat{\alpha}_k, 1\}$，置$x^{k+1} = x^k + \alpha_k d^k$。若$\alpha_k = \hat{\alpha}_k$,则置$I(x^{k+1}) = I(x^k) + \{p\}$;否则置$I(x^{k+1}) = I(x^k)$，置$k := k + 1$，转步骤（2）。

5.2 实验结果

利用$numpy.random.multivariate_normal(mean, cov, point_number)$函数生成数据，将各50个点，分别以mean为[-1,2]和[1,-1]作为第一类数据集，mean为[5,-5]和[-6,2]各50个点作为第二类数据集，其中cov矩阵均为$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$，将两类型数据各取10% 的数据作为检测数据，剩下数据都拿来作为训练数据。实验结果如下图示。



其中红色和黄色分别表示两种类型的数据，加粗加绿的点是支持向量，共有5个点，因为采用了高斯核函数，所以SVM的分类超平面不是直线，图中的加粗的轮廓线就是分类超平面。用于测试点20个点都用SVM分类器分类正确了。

参考文献

[1] 《机器学习》，周志华，清华大学出版社，2016年版

[2] Madsen K, Nielsen H B, Tingleff O. Methods for non-linear least squares problems[J]. 2004.

[3] 《最优化基础理论与方法》，王燕军，梁治安，复旦大学出版社，2011年版