机器学习导论
习题四

141242006, 袁帅, 141242006@smail.nju.edu.cn

2017 年 5 月 14 日

1 [20pts] Reading Materials on CNN

卷积神经网络 (Convolution Neural Network, 简称 CNN) 是一类具有特殊结构的神经网络, 在深度学习的发展中具有里程碑式的意义。其中, Hinton 于 2012 年提出的AlexNet可以说是深度神经网络在计算机视觉问题上一次重大的突破。

关于 AlexNet 的具体技术细节总结在经典文章"ImageNet Classification with Deep Convolutional Neural Networks", by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton in NIPS'12, 目前已逾万次引用。在这篇文章中, 它提出使用 ReLU 作为激活函数, 并创新性地使用 GPU 对运算进行加速。请仔细阅读该论文, 并回答下列问题 (请用 1-2 句话简要回答每个小问题, 中英文均可)。

(a) **[5pts]** Describe your understanding of how ReLU helps its success? And, how do the GPUs help out?

(b) **[5pts]** Using the average of predictions from several networks help reduce the error rates. Why?

(c) **[5pts]** Where is the dropout technique applied? How does it help? And what is the cost of using dropout?

(d) **[5pts]** How many parameters are there in AlexNet? Why the dataset size(1.2 million) is important for the success of AlexNet?

关于 CNN, 推荐阅读一份非常优秀的学习材料, 由南京大学计算机系吴建鑫教授[1]所编写的讲义 Introduction to Convolutional Neural Networks[2], 本题目为此讲义的 Exercise-5, 已获得吴建鑫老师授权使用。

[1] 吴建鑫教授主页链接为cs.nju.edu.cn/wujx

[2] 由此链接可访问讲义https://cs.nju.edu.cn/wujx/paper/CNN.pdf

Solution.

(a) ReLU is used as the neuron's activation function. The advantages of ReLU could be summarized as: (i) to provide non-saturating nonlinearity and hence to make the model fit complicated functions better; (ii) to speed up the training session; (iii) to prevent gradient from vanishing in back-propagation under the SGD scheme.

GPUs facilitate the training session by optimizing computational costs of 2D convolution. Specifically, the layer training memory are distributed separately on two GPUs and the GPUs only communicate on certain layers.

(b) Averaging predictions from several networks is a perfect example of *Ensemble Learning*. In AlexNet, various random factors are introduced in the model by random initiations and random dropout processes, so the learning result could be diverse. By averaging random variables, we can make occasional errors less critical.

(c) Dropout is applied in the first two fully-connected layers by setting to zero the output of each hidden neuron with probability 0.5. This technique reduces complex co-adaptations of neurons, forces neurons to learn robust features and therefore reduces overfitting. Nevertheless, the running time with dropout technique is doubled because more iterations are required to converge.

(d) This model's parameters include convolutional kernels, neuron biases and fully-connected weights. The number of parameters in AlexNet is estimated as around 60 million. A large dataset is necessary for complicated models to reduce overfitting.

2 [20pts] Kernel Functions

(1) 试通过定义证明以下函数都是一个合法的核函数：

(i) **[5pts]** 多项式核: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j)^d$;

(ii) **[10pts]** 高斯核： $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$, 其中 $\sigma > 0$.

(2) **[5pts]** 试证明 $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1+e^{-\mathbf{x}_i^T\mathbf{x}_j}}$ 不是合法的核函数。

Proof.

(1). By definition, in order to prove the kernel function's validity, we try to find a mapping $\boldsymbol{\phi} : \mathbb{R}^n \mapsto \mathbb{R}^m$, such that $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{y})$.

Polynomial kernel: Wikipedia[1] gives a specific proof for the case $d = 2$. To expand that proof, for any vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^T$, consider

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})^d = (\sum_{i=1}^{n} x_i y_i)^d = \sum_{\substack{0 \le t_1, t_2, \cdots, t_n \le d \\ t_1 + t_2 + \cdots + t_n = d}} \left[\binom{d}{t_1, t_2, \cdots, t_n} \cdot \prod_{j=1}^{n} (x_j y_j)^{t_j} \right], \quad (2.1)$$

in which $\binom{d}{t_1, t_2, \cdots, t_n} = \binom{d}{t_1}\binom{d-t_1}{t_2}\binom{d-t_1-t_2}{t_3}\cdots\binom{d-t_1-\cdots-t_{n-1}}{t_n}$ is the *multinomial coefficient*. Suppose $\boldsymbol{t} = (t_1, t_2, \cdots, t_n)^{\mathrm{T}}$ is an assignment such that $0 \leq t_i \leq d$ for all $i \in \{1, 2, \cdots, n\}$ and $\sum_{i=1}^{n} t_i = d$. We can define function

$$f(\boldsymbol{x}, \boldsymbol{t}) = \sqrt{\binom{d}{t_1, t_2, \cdots, t_n}} \cdot \prod_{i=1}^{n} x_i^{t_i}, \tag{2.2}$$

so that $f(\boldsymbol{x}, \boldsymbol{t})f(\boldsymbol{y}, \boldsymbol{t})$ is equivalent to each term in the summation of Eq.(2.1). In fact, numerous ($m = \binom{d+n-1}{d}$ in total) distinct $\boldsymbol{t}$ can be found, which can be enumerated as $\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(m)}$. Thus we can define

$$\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x}) = \left(f(\boldsymbol{x}, \boldsymbol{t}^{(1)}), f(\boldsymbol{x}, \boldsymbol{t}^{(2)}), ..., f(\boldsymbol{x}, \boldsymbol{t}^{(n)})\right)^{\mathrm{T}}, \tag{2.3}$$

so that

$$\begin{aligned}
\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{y}) &= \sum_{i=1}^{m} f(\boldsymbol{x}, \boldsymbol{t}^{(i)})f(\boldsymbol{y}, \boldsymbol{t}^{(i)}) = \sum_{i=1}^{m}\left[\binom{d}{t_1^{(i)}, t_2^{(i)}, \cdots, t_n^{(i)}} \cdot \prod_{j=1}^{n}(x_j y_j)^{t_j^{(i)}}\right] \\
&= \sum_{\substack{0 \leq t_1, t_2, \cdots, t_n \leq d \\ t_1+t_2+\cdots+t_n=d}}\left[\binom{d}{t_1, t_2, \cdots, t_n} \cdot \prod_{j=1}^{n}(x_j y_j)^{t_j}\right] = \kappa(\boldsymbol{x}, \boldsymbol{y}).
\end{aligned} \tag{2.4}$$

Therefore, the polynomial kernel function can be construed as a dot product in a $\binom{d+n-1}{d}$-dimensional space, so the corresponding polynomial kernel function is valid.

RBF kernel: According to [2], we can taylor expand the kernel function as

$$\begin{aligned}
\kappa(\boldsymbol{x}, \boldsymbol{y}) &= \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}) = \exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2}) \cdot \exp(\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}}{\sigma^2}) \cdot \exp(-\frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}{2\sigma^2}) \\
&= \exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2}) \cdot \sum_{d=0}^{\infty}\frac{(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})^d}{d! \cdot \sigma^{2d}} \cdot \exp(-\frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}{2\sigma^2}) \\
&= \sum_{d=0}^{\infty}\frac{\exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2})}{\sqrt{d!}\sigma^d} \cdot (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})^d \cdot \frac{\exp(-\frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}{2\sigma^2})}{\sqrt{d!}\sigma^d} \\
&= \sum_{d=0}^{\infty}\frac{\exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2})}{\sqrt{d!}\sigma^d} \cdot \boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x}, d)^{\mathrm{T}}\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{y}, d) \cdot \frac{\exp(-\frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}{2\sigma^2})}{\sqrt{d!}\sigma^d}.
\end{aligned} \tag{2.5}$$

Hence, by defining vector function $\boldsymbol{g}(\boldsymbol{x}, d) = \frac{\exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2})}{\sqrt{d!}\sigma^d} \cdot \boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x}, d)$,[3] we have mapping $\boldsymbol{\phi}_{\mathbf{rbf}} : \mathbb{R}^n \mapsto \mathbb{R}^\infty$, which concatenates all $\boldsymbol{g}(\boldsymbol{x}, d)$, i.e.

$$\boldsymbol{\phi}_{\mathbf{rbf}}(\boldsymbol{x}) = (\boldsymbol{g}(\boldsymbol{x}, 0)^{\mathrm{T}}, \boldsymbol{g}(\boldsymbol{x}, 1)^{\mathrm{T}}, \boldsymbol{g}(\boldsymbol{x}, 2)^{\mathrm{T}}, \cdots)^{\mathrm{T}}, \tag{2.6}$$

[3]When $d = 0$, $\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x}, d)$ is the scalar 1; otherwise, it is the mapping for polynomial kernel with parameter d.

so that

$$\begin{aligned}\boldsymbol{\phi}_{\mathbf{rbf}}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}_{\mathbf{rbf}}(\boldsymbol{y}) &= \sum_{d=0}^{\infty}\boldsymbol{g}(\boldsymbol{x},d)^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{y},d)\\ &= \sum_{d=0}^{\infty}\frac{\exp(-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2})}{\sqrt{d!}\sigma^d}\cdot\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{x},d)^{\mathrm{T}}\boldsymbol{\phi}_{\mathbf{poly}}(\boldsymbol{y},d)\cdot\frac{\exp(-\frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}{2\sigma^2})}{\sqrt{d!}\sigma^d}\\ &= \kappa(\boldsymbol{x},\boldsymbol{y}).\end{aligned} \tag{2.7}$$

Therefore, the RBF kernel function can be rewritten as a dot product in an infinite-dimensional space, so the corresponding kernel function is valid.

(2). A counter-example could be raised to disprove the kernel function's validity: consider scalar (unidimensional vector) $x_1 = \frac{1}{2}$ and $x_2 = 10$. The kernel matrix

$$\boldsymbol{K} = \begin{bmatrix}\kappa(x_1,x_1) & \kappa(x_1,x_2)\\ \kappa(x_2,x_1) & \kappa(x_2,x_2)\end{bmatrix} = \begin{bmatrix}\frac{1}{1+\exp(-\frac{1}{4})} & \frac{1}{1+\exp(-5)}\\ \frac{1}{1+\exp(-5)} & \frac{1}{1+\exp(-100)}\end{bmatrix}. \tag{2.8}$$

For non-zero vector $\boldsymbol{w} = (-\frac{1}{1+\exp(-5)}, \frac{1}{1+\exp(-\frac{1}{4})})$, the quadratic form $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{w}$
$= \frac{1}{1+\exp(-\frac{1}{4})}(-\frac{1}{1+\exp(-5)})^2 - 2\frac{1}{1+\exp(-5)}\cdot\frac{1}{1+\exp(-5)}\cdot\frac{1}{1+\exp(-\frac{1}{4})} + \frac{1}{1+\exp(-100)}(\frac{1}{1+\exp(-\frac{1}{4})})^2$
$= \frac{1}{1+\exp(-\frac{1}{4})}\left[\frac{1}{1+\exp(-100)}\cdot\frac{1}{1+\exp(-\frac{1}{4})} - (\frac{1}{1+\exp(-5)})^2\right] < 0$. Therefore, $\boldsymbol{K}$ is not positive semi-definite, and the kernel function is not valid. □

3 [25pts] SVM with Weighted Penalty

考虑标准的 SVM 优化问题如下 (即课本公式 (6.35)),

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i}\quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i\\ \text{s.t.}\quad & y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1-\xi_i\\ & \xi_i \geq 0, i = 1,2,\cdots,m.\end{aligned} \tag{3.1}$$

注意到，在(3.1)中，对于正例和负例，其在目标函数中分类错误的"惩罚"是相同的。在实际场景中，很多时候正例和负例错分的"惩罚"代价是不同的，比如考虑癌症诊断，将一个确实患有癌症的人误分类为健康人，以及将健康人误分类为患有癌症，产生的错误影响以及代价不应该认为是等同的。

现在，我们希望对负例分类错误的样本 (即 false positive) 施加 $k > 0$ 倍于正例中被分错的样本的"惩罚"。对于此类场景下，

(1) **[10pts]** 请给出相应的 SVM 优化问题;

(2) **[15pts]** 请给出相应的对偶问题，要求详细的推导步骤，尤其是如 KKT 条件等。

Solution.

(1). Similarly, we introduce slack variable ξ_i to measure the loss. In the standard form, both false positive and false negative cases have the same contribution to total loss; in this

problem, we distinguish these cases and assign different penalties separately. Specifically, we can write each *penalty* as

$$\begin{aligned}\ell_i &= \mathbb{I}(y_i = 1) \cdot C\xi_i + \mathbb{I}(y_i = -1) \cdot kC\xi_i \\ &= \frac{y_i + 1}{2} \cdot C\xi_i + \frac{-y_i + 1}{2} \cdot kC\xi_i \\ &= \frac{C}{2}\xi_i[k + 1 + (1 - k)y_i]. \end{aligned} \tag{3.2}$$

Note that in Eq.(3.2), the denominator 2 can be absorbed by constant C. Therefore, the formalized optimization problem could be presented as

$$\begin{aligned}\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i[k + 1 + (1 - k)y_i] \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, 2, \cdots, m. \end{aligned} \tag{3.3}$$

(2). The Lagrange function is

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i[k+1+(1-k)y_i] + \sum_{i=1}^{m}\lambda_i(1-\xi_i-y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)) - \sum_{i=1}^{m}\mu_i\xi_i. \tag{3.4}$$

By setting derivatives $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$ and $\frac{\partial L}{\partial \boldsymbol{\xi}} = 0$, we obtain

$$\boldsymbol{w} = \sum_{i=1}^{m}\lambda_i y_i \boldsymbol{x}_i, \tag{3.5}$$

$$\sum_{i=1}^{m}\lambda_i y_i = 0, \tag{3.6}$$

$$\lambda_i + \mu_i = C[k + 1 + (1 - k)y_i]. \tag{3.7}$$

Plugging back Eq.(3.5), Eq.(3.6) and Eq.(3.7), we get the dual problem as

$$\begin{aligned}\max_{\boldsymbol{\lambda}} \quad & \sum_{i=1}^{m}\lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{m}\lambda_i y_i = 0, \\ & 0 \leq \lambda_i \leq C[k + 1 + (1 - k)y_i],\ i = 1, 2, \cdots, m. \end{aligned} \tag{3.8}$$

and the corresponding KKT conditions are: for each $i = 1, 2, \cdots, m$,

$$\begin{cases} \lambda_i \geq 0, \quad \mu_i \geq 0, \\ 1 - \xi_i - y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \leq 0, \\ \lambda_i(1 - \xi_i - y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)) = 0, \\ \xi_i \geq 0, \quad \mu_i\xi_i = 0. \end{cases} \tag{3.9}$$

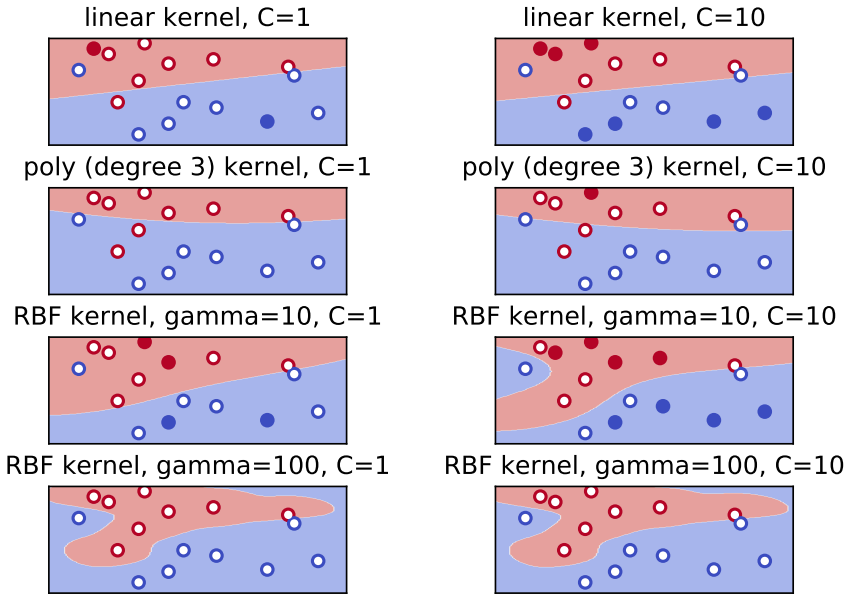
4 [35pts] SVM in Practice - LIBSVM

支持向量机 (Support Vector Machine，简称 SVM) 是在工程和科研都非常常用的分类学习算法。有非常成熟的软件包实现了不同形式 SVM 的高效求解，这里比较著名且常用的如 LIBSVM[4]。

(1) **[20pts]** 调用库进行 SVM 的训练，但是用你自己编写的预测函数作出预测。

(2) **[10pts]** 借助我们提供的可视化代码，简要了解绘图工具的使用，通过可视化增进对 SVM 各项参数的理解。详细编程题指南请参见链接：`http://lamda.nju.edu.cn/ml2017/PS4/ML4_programming.html`.

(3) **[5pts]** 在完成上述实践任务之后，你对 SVM 及核函数技巧有什么新的认识吗？请简要谈谈。



Solution. Above are the results of six different kernel configurations on the demo dataset. All support vectors are marked by a white dot. To illustrate how different kernels work in SVM, I tried some more interesting data points, which are shown in Fig.(1) and Fig.(2).

In the XOR problem (Fig.(1)), data from the same class are put on one of the two diagonal areas. In this case, linear kernel is not functioning: it simply interpret all data points as positive instances, so the visualization method based on contour doesn't work. The cubic polynomial kernel can only deal with one part of the positive cases, while leaving the other half mistaken. Contrarily, RBF kernel works perfectly, dividing the input space into four distinct parts. Moreover, when we raise the parameters gamma and C in the RBF model, more data end up support vectors, embodying a growing model robustness.

[4]LIBSVM 主页课参见链接：`https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

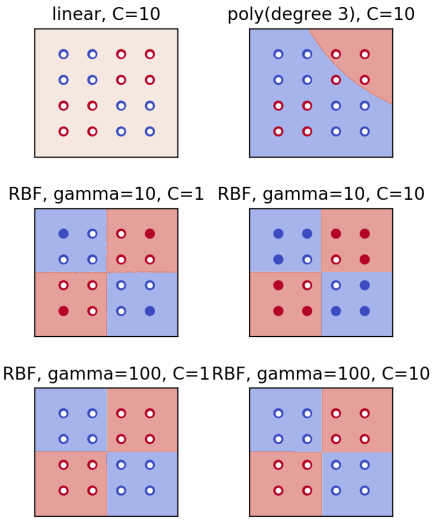


Figure 1: The XOR problem

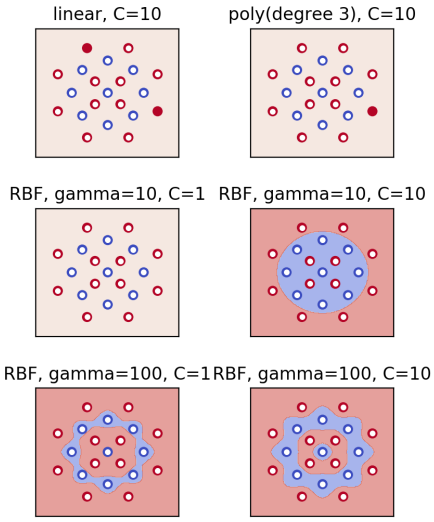


Figure 2: The embedded rings problem

In the embedded rings problem (Fig.(refring)), certain data are distributed as a ring, and rings of different classes are embedded one another. In this complicated scenario, linear and polynomial kernels, just as we expected, doesn't work. A simple RBF kernel implementation also fails by labelling all data as the same class. When we try to increase the RBF parameters, the results are apparently getting better by separating the rings apart.

Reference

[1] Wikipedia. *Polynomial kernel*. `https://en.wikipedia.org/wiki/Polynomial_kernel`.

[2] Matthew Bernstein. *The Radical Basis Function Kernel*. University of Wisconsin - Madison. `http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf`.

[3] Scikit-learn. *sklearn.svm.SVC Documentation*. `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.