机器学习导论
习题四

Boardwell, Nanjing University

2017 年 7 月 10 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。

Solution.
(a):
ReLu: Comparing to sigmoid function，ReLu won't have the problem of gradient saturated

[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`

in sigmoid which will "kill gradient" in condition of large value of x (which means the gradient will be so small that the parameters will never actually be updated). ReLu function will keep the gradient being 1 as long as x being in position region and won't have the problem of too samll gradient. However, ReLu will sitll have the problem of not updated when most independent variable being negative which called "Dead ReLu".
ReLu function is easy to calculate its gradient which is always 1, comparing to the gradient of sigmoid that have to calculate value of exponent which cost a lot of time.ReLu converges much faster than sigmoid for its gradient being relatively big than sigmoid in most condition.
GPU: The most common calculate unit, CPU, is good at processing various complex problems but GPU is good at processing larger scale simple problems like millions of addition or subtraction for its initial function to image rendering. Problems in CNN is mostly simple but have very large scale.
(b):
Because one single trained CNN's parameters will always have the problem of overfitting and accidental error. Averaging predictions from several networks helps to reduce the accidental error and is more likely to get near the optimal condition.
Considering that the target funciton to be minimized have a rapidly changed value in value space, all of networks are eager to reach the minimum but have accidental errors(overfitting). The average of their predictions are likely to being more optimal.
(c):
Dropout is applied in full-connected layers. It helps to improve the problem of overfitting by randomly cancel two neuron's connect. Actually the skill of dropout trains a large ensemble of models that share similiar parameters and take the average of them. Just like explanation to question (b), average of predictions helps to improve performance.
(d):
There are 60 million parameters in AlexNet.
Large network have the better performance to approximate the solution so we have to use it. Large size of dataset helps to fully train so many parameters and reduce problem of overfitting because using small size of dataset to train a large network will lead to serious overfitting and large dataset have so many information of this problem that we can believe it represents the fact. So the large size of dataset is important for AlexNet.

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): Considering the condition of linear kernel function, which is $k_1(x_i, x_j) = x_i^T w_j$.
The kernel function can be seperated as $k(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ in which $\phi(x_i) = x_i$. So that we can calculate the value of $z^T Kz$ and prove that it is always positive as follows.

$$\begin{aligned}
z^T Kz &= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i K_{ij} z_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i \phi(x_i)^T \phi(x_j) z_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i x_i^T x_j z_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i \sum_{k=1}^{d}(x_{i_k}^T x_{j_k}) z_j \\
&= \sum_{k=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} z_i x_{i_k}^T x_{j_k} z_j \\
&= \sum_{k=1}^{d}(\sum_{i=1}^{m} z_i x_{i_k})^2 \\
&\geq 0
\end{aligned} \tag{2.1}$$

So that linear kernel $k_1(x_i, x_j) = x_i^T x_j$ has been proved as legal kernel function.
(i): After get linear kernel $k_1(x_i, x_j) = x_i^T x_j$ as a legal kernel function, we can use the property of kernel function to prove the polynomial kernel as legal.
According to equation (6.26) in textbook Machine Learning, two kernel functions' direct product is also a legal kernel function.
$k_2(x_i, x_j) = k_1(x_i, x_j)k_1(x_i, x_j) = (x_i^T x_j)^2$ (Because $x_i^T x_j$ is not a matrix or vector, the position of $x_i^T x_j$ can be moved arbitrary).
Repeatly use this property, we can prove that any positive integer times of linear kernel function is also a legal kernel function.
(ii): Now that we has know polynomial kernel functions are all legal, we can use Taylor unfolds to prove Gaussian kernel function is also legal.
Unfold the quadratic term :

$$||x_i - x_j||^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$$

So that

$$k(x_i, x_j) = e^{-\frac{x_i^T x_i}{2\sigma^2}} e^{-\frac{x_j^T x_j}{2\sigma^2}} e^{\frac{x_i^T x_j}{\sigma^2}}$$

in which

$$e^{\frac{x_i^T x_j}{\sigma^2}} = \sum_{n=0}^{\infty} \frac{1}{n!}\frac{(x_i^T x_j)^n}{\sigma^{2n}} = \sum_{n=0}^{\infty} \frac{1}{\sqrt{n!}\sigma^n}(x_i^T x_j)^n \frac{1}{\sqrt{n!}\sigma^n}$$

So that use the property (6.27) in our textbook, give $g(x) = \frac{1}{\sqrt{n!}\sigma^n}$, $k_n(x_i, x_j) = (x_i^T x_j)^n$, then we can prove that each item of $e^{\frac{x_i^T x_j}{\sigma^2}}$ is a legal kernel function. So the linear combination of each item, which is $e^{\frac{x_i^T x_j}{\sigma^2}}$ is also a legal kernel function.
At last, use $g(x) = e^{-\frac{x^T x}{2\sigma^2}}$, according to (6.27), $k_{RBF}(x_i, x_j) = g(x_i)e^{\frac{x_i^T x_j}{\sigma^2}}g(x_j)$ is a legal kernel function.
(2):
Assume that the input space is R^d and we can structure a counterexample. Let $x_1 = (1; 1)$, $x_2 = (3; 3)$, the kernel matrix can be calculated as :

$$K = \begin{bmatrix} \frac{1}{1+e^{-2}} & \frac{1}{1+e^{-6}} \\ \frac{1}{1+e^{-6}} & \frac{1}{1+e^{-18}} \end{bmatrix} \tag{2.2}$$

Let $z = (1; -1)$:

$$z^T K z = \frac{1}{1+e^{-2}} + \frac{1}{1+e^{-18}} - \frac{2}{1+e^{-6}} = -0.1142 < 0$$

So it is not a legal kernel function. □

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(**??**)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): SVM optimal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i,\eta_i} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^m \xi_i + kC\sum_{i=1}^m \eta_i \\ \text{s.t.} \quad & \mathbf{w}^\mathrm{T}\mathbf{x}_i + b \geq 1 - \xi_i, \quad y_i = 1; \\ & \mathbf{w}^\mathrm{T}\mathbf{x}_i + b \leq -1 + \eta_i, \quad y_i = -1; \\ & \xi_i \geq 0, \quad \eta_i \geq 0 \end{aligned} \tag{3.2}$$

(2):

$$\begin{aligned}
L(\mathbf{w}, b, \alpha, \beta, \xi, \eta, \mu, \tau) = &\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i + kC\sum_{i=1}^{m}\eta_i \\
&+ \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)(1 - \xi_i - (\mathbf{w}^T x_i + b)) + \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)(1 + -\eta_i + (\mathbf{w}^T x_i + b)) \\
&- \sum_{i=1}^{m}\mu_i\xi_i - \sum_{i=1}^{m}\tau_i\eta_i
\end{aligned} \tag{3.3}$$

Let $L(\mathbf{w}, b, \alpha, \beta, \xi, \eta, \mu, \tau)$'s partial derivative to $\mathbf{w}$, b, ξ, η equals to 0:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)x_i + \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)x_i \\
\frac{\partial L}{\partial b} &= -\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1) + \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1) \\
\frac{\partial L}{\partial \xi_i} &= C - \alpha_i\mathbb{I}(y_i = 1) + \mu_i \\
\frac{\partial L}{\partial \eta_i} &= kC - \beta_i\mathbb{I}(y_i = -1) - \tau_i
\end{aligned} \tag{3.4}$$

So that the variables have such relationship:

$$\begin{aligned}
\mathbf{w} &= \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)x_i - \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)x_i \\
\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1) &= \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1) \\
C &= \alpha_i\mathbb{I}(y_i = 1) - \mu_i \\
kC &= \beta_i\mathbb{I}(y_i = -1) + \tau_i
\end{aligned} \tag{3.5}$$

So that the lagrange function can be simplified as:

$$\begin{aligned}
L =&\frac{1}{2}(\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)x_i - \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)x_i)^2 + \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1) + \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1) \\
&- \sum_{s=1}^{m}\alpha_s\mathbb{I}(y_s = 1)(\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)x_i^T x_s - \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)x_i^T x_s) \\
&+ \sum_{s=1}^{m}\beta_s\mathbb{I}(y_s = -1)(\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1)x_i^T x_s - \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1)x_i^T x_s) \\
=&\sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i = 1) + \sum_{i=1}^{m}\beta_i\mathbb{I}(y_i = -1) \\
&+ \mathbf{x}_i^T\mathbf{x}_j[-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\mathbb{I}(y_i = 1)\alpha_j\mathbb{I}(y_j = 1) - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\beta_i\mathbb{I}(y_i = -1)\beta_j\mathbb{I}(y_j = -1) \\
&+ \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\mathbb{I}(y_i = 1)\beta_j\mathbb{I}(y_j = -1)]
\end{aligned} \tag{3.6}$$

The dual problem is:

$$\begin{aligned}
\max_{\alpha,\beta} \quad & \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i=1)+\sum_{i=1}^{m}\beta_i\mathbb{I}(y_i=-1) \\
& +\frac{1}{2}[\sum_{i=1}^{m}\sum_{j=1}^{m}(-\alpha_i\mathbb{I}(y_i=1)\alpha_j\mathbb{I}(y_j=1)-\beta_i\mathbb{I}(y_i=-1)\beta_j\mathbb{I}(y_j=-1)+\alpha_i\mathbb{I}(y_i=1)\beta_j\mathbb{I}(y_j=-1)]\mathbf{x}_i^T\mathbf{x}_j \\
\text{s.t.} \quad & \sum_{i=1}^{m}\alpha_i\mathbb{I}(y_i=1)-\sum_{i=1}^{m}\beta_i\mathbb{I}(y_i=-1)=0 \\
& 0\le\alpha_i\mathbb{I}(y_i=1)\le C, i=1,2,...m. \\
& 0\le\beta_i\mathbb{I}(y_i=1)\le kC, i=1,2,...m.
\end{aligned} \tag{3.7}$$

KKT condition:

$$\begin{cases}
\alpha_i\ge 0, \quad \beta_i\ge 0; \\
\mu_i\ge 0, \quad \tau_i\ge 0; \\
\xi_i\ge 0, \quad \eta_i\ge 0; \\
\mu_i\xi_i=0, \quad \tau_i\eta_i=0; \\
\quad f(\mathbf{x}_i)-1+\xi_i\ge 0, \quad y_i=1; \\
\alpha(f(\mathbf{x}_i)-1+\xi_i)=0 \\
\quad f(\mathbf{x}_i)-1+\eta_i\ge 0, \quad y_i=-1; \\
\beta(f(\mathbf{x}_i)-1+\eta_i)=0
\end{cases}$$

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[3]。

(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。

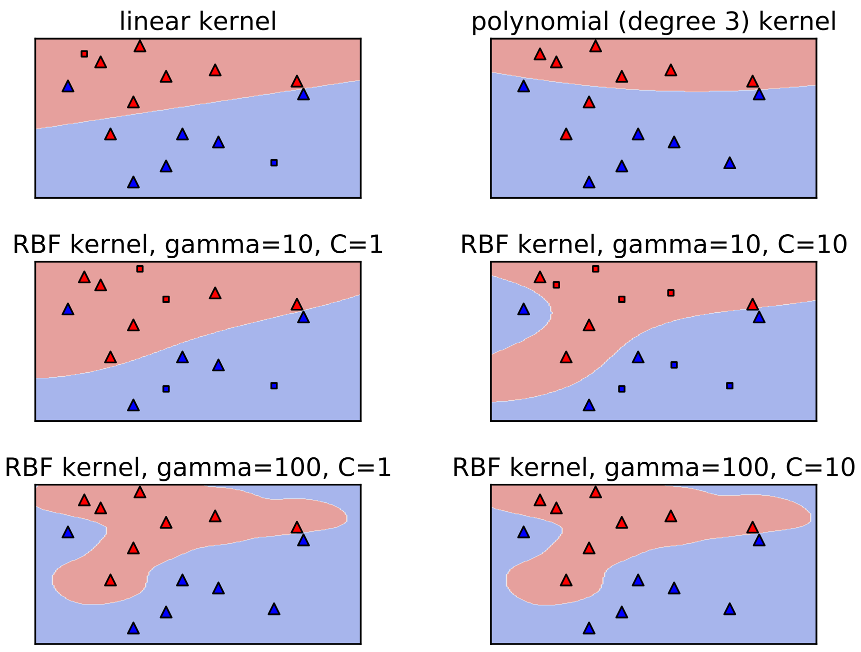
(2) **[10pts]** 借助我们提供的可视化代码，简要了解绘图工具的使用，通过可视化增进对SVM各项参数的理解。详细编程题指南请参见链接：`http://lamda.nju.edu.cn/ml2017/PS4/ML4_programming.html`.

(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.

Figure of support vectors are as follows:

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



Big triangles are support vectors and little squares are normal vectors.

(3):

SVM is effective to most classification problems and relatively have much fewer parameters than neural networks but still works good. SVM's mathematical explanation is relatively clear, comparing to convolutional neural network being a "black-box" in which no one knows what there are.

RBF kernel (Gaussian kernel) is useful in most condition and can always get results not bad. Kernel tricks enable us to classify samples in a higher demention which is very important to solve problems.