caffe中的矩阵乘法

无论在全连接神经网络还是在卷积神经网络的计算中，矩阵乘法是非常重要的一块，矩阵乘法的快慢会直接影响到神经网络训练以及推断的快慢。在这一小节中我们主要是学习一下caffe中的矩阵乘法函数的使用，因为caffe中的所有的数据都是存储在连续的一维内存当中的，这个和matlab或者python的矩阵表示形式不太一样，所以感觉要单独说明一下这个矩阵乘法API的调用形式。

caffe_cpu_gemm函数

```
template<>   //模板偏特化，针对float数据类型
    void caffe_cpu_gemm<float>(const CBLAS_TRANSPOSE TransA,
            const CBLAS_TRANSPOSE TransB, const int M, const int N, const int K,
            const float alpha, const float* A, const float* B, const float beta,
            float* C) {
        int lda = (TransA == CblasNoTrans) ? K : M;
        int ldb = (TransB == CblasNoTrans) ? N : K;
        cblas_sgemm(CblasRowMajor, TransA, TransB, M, N, K, alpha, A, lda, B,
                ldb, beta, C, N);
    }
template<>  //模板偏特化，正对double数据类型
    void caffe_cpu_gemm<double>(const CBLAS_TRANSPOSE TransA,
            const CBLAS_TRANSPOSE TransB, const int M, const int N, const int K,
            const double alpha, const double* A, const double* B, const double beta,
            double* C) {
        int lda = (TransA == CblasNoTrans) ? K : M;
        int ldb = (TransB == CblasNoTrans) ? N : K;
        cblas_dgemm(CblasRowMajor, TransA, TransB, M, N, K, alpha, A, lda, B,
                ldb, beta, C, N);
    }
```

上述代码使用了模板偏特化技术，最主要的原因是float类型和double类型的矩阵乘法调用的底层函数库不一样。

该函数所对应的数学公式为：

$$\boldsymbol{C} \leftarrow \alpha * op(\boldsymbol{A}) op(\boldsymbol{B}) + \beta \boldsymbol{C} \tag{1}$$

这里的op代表的意思是是否进行了转置，那我们还得看一下矩阵的维度，这三个矩阵的维度分别为：

$$op(\boldsymbol{A}) \in (m, k) \tag{2}$$

$$op(\boldsymbol{B}) \in (k, n) \tag{3}$$

$$\boldsymbol{C} \in (m, n) \tag{4}$$

举一个例子，如果我们的$\boldsymbol{A}$矩阵维度为$(3, 2)$，$\boldsymbol{B}$矩阵的维度为$(3, 2)$，而我们的计算式为：

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}^T \tag{5}$$

其中$\boldsymbol{A}=\boldsymbol{B}$，其中$\boldsymbol{A}$的值为：

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{pmatrix} \tag{6}$$

$\boldsymbol{B}^T$的值为：

$$\begin{pmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{pmatrix} \tag{7}$$

那么$\boldsymbol{C}$的结果为：

$$\begin{pmatrix} 1 & 3 & 5 \\ 3 & 13 & 23 \\ 5 & 23 & 41 \end{pmatrix} \tag{8}$$

那我们输入的m为3，k为2，n为3。**切记：m，n，k是描述经过转置之后矩阵的维度，不是转置之前矩阵的维度。**

那相应的的函数调用为：

```
caffe_cpu_gemm(CblasNoTrans,CblasTrans,
                      3, 3, 2,
            1, A, B, 1,
            C) ;
```

caffe_cpu_gemv函数

```
template <typename Dtype>
void caffe_cpu_gemv(const CBLAS_TRANSPOSE TransA, const int M, const int N,
    const Dtype alpha, const Dtype* A, const Dtype* x, const Dtype beta,
    Dtype* y);
```

该函数所对应的数学公式为：

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{A}\boldsymbol{x} + \beta \boldsymbol{x} \tag{9}$$

维度对应关系为：

$$\boldsymbol{A} \in (m, n) \tag{10}$$

向量$\boldsymbol{x}$和$\boldsymbol{y}$的维度是一样的，维度有两种可能性：

$$\boldsymbol{x}, \boldsymbol{y} \in (m, 1)\text{或者}(n, 1) \tag{11}$$

但注意向量都是列向量。

另外gemv函数和gemm函数使用起来有点差别，m，n是矩阵$\boldsymbol{A}$的维度，而不是$\boldsymbol{A}^T$的维度，这个和gemm的不一样，需要注意。