# 如何在C++中构建一个像TysFoe流的神经网络框架：第1部分

好的，正如我在这组教程中所讨论的，我将展示如何构建一个类似于Tensorflow的神经网络框架，但是在C++中。

本教程将分为编码最小可行的产品：

1. 成本函数
2. 最小化成本函数的极小化子
3. 神经网络模型及其反向传播

在本教程中，我们将讨论成本函数。

首先，让我们看看整个类的头现在的样子，并逐行解释它：

有很多东西要盖，我们开始吧。

第1-19行只是Shogun这样的库中的常规代码。它包括Stan的头（我们在教程的第0部分中讨论过）以及EGIGN（它是C++中的线性代数库）。

第21-28行定义了一些类型，以便以后更容易引用。例如，StanVector是一个特征矩阵，其中包含一组stan变量（有关详细信息，请参见第0部分）。

然后，我们有一个主要类：StanFirstOrderSAGCostFunction，它为定义随机平均梯度成本函数提供了一个接口。它还提供了函数get\_gradient和get\_average\_gradient，这是大部分工作要做的地方。

但是在我解释这是如何工作的之前，我们需要看看StanFirstOrderSAGCostFunction的成员。

第一个成员是m\_X，m\_y，基本上是成本函数的训练数据和标签。

m\_trainable\_参数顾名思义是一组stan变量，它们是成本函数的参数。

m\_cost\_for\_ith\_point和m\_total\_cost同样是顾名思义的函数，用于评估第i个点相对于可训练参数的误差，以及相对于第i个数据点的所有成本的总成本。

所以，当调用get\_gradient（）时，我们使用m\_total\_cost的定义计算当前参数的误差，然后使用stan得到这个误差代价函数相对于每个可训练参数的梯度，这就是stan的能力！有关实现的详细信息，请查看我在其中实现类的链接：

With this class done, we can now define any arbitrary cost function in terms of stan, and calculate the gradient of it with respect to all parameters using stan. For an example of how to use the class, checkout an example of it being used here, where I implemented mean squared error:

### A Word on Shogun

In case you’re interested in joining Open Source, Shogun is a great place to start. They have a super supportive community, and they welcome new comers to Opensource, so swing by and see if you can help with some of the issues labelled “beginner friendly” on their github!