**Understanding Hinton’s Capsule Networks. Part III: Dynamic Routing Between Capsules.**

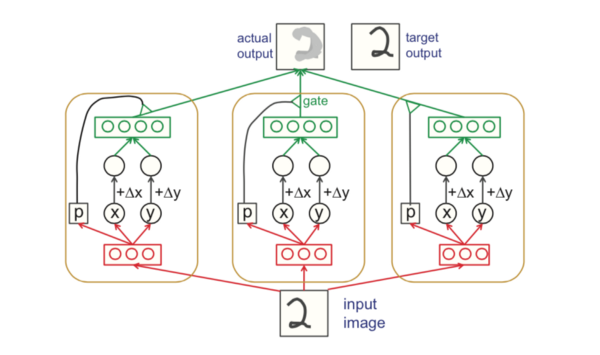
关于我们新出版物[AI³](https://medium.com/ai%C2%B3-theory-practice-business?source=logo-bc670b9a1aca---ab837e78463f)的快速公告。我们一起找最好的作家来谈论人工智能和机器学习的理论，实践和商业。跟随它，保持对AI最新的趋势的关注。

Introduction

This is the third post in the series about a new type of neural network, based on capsules, called CapsNet. I already talked about the intuition behind it, as well as what is a capsule and how it works. In this post, I will talk about the novel dynamic routing algorithm that allows to train capsule networks.

介绍

这是关于基于胶囊的新型神经网络系列（名为CapsNet）的第三篇文章。 我已经谈到了它背后的灵感，什么是胶囊和它是如何工作的。 在这篇文章中，我将讨论允许训练胶囊网络的新型动态路由算法。



One of the earlier figures explaining capsules and routing between them. [Source](http://helper.ipam.ucla.edu/publications/gss2012/gss2012_10754.pdf).

这是早期的一张图片，解释了“胶囊”网络和它们之间的路由。[[图片来源](http://helper.ipam.ucla.edu/publications/gss2012/gss2012_10754.pdf)]

As I showed in Part II, a capsule i in a lower-level layer needs to decide how to send its output vector to higher-level capsules j. It makes this decision by changing scalar weight c\_ij that will multiply its output vector and then be treated as input to a higher-level capsule. Notation-wise, c\_ij represents the weight that multiplies output vector from lower-level capsule i and goes as input to a higher level capsule j.

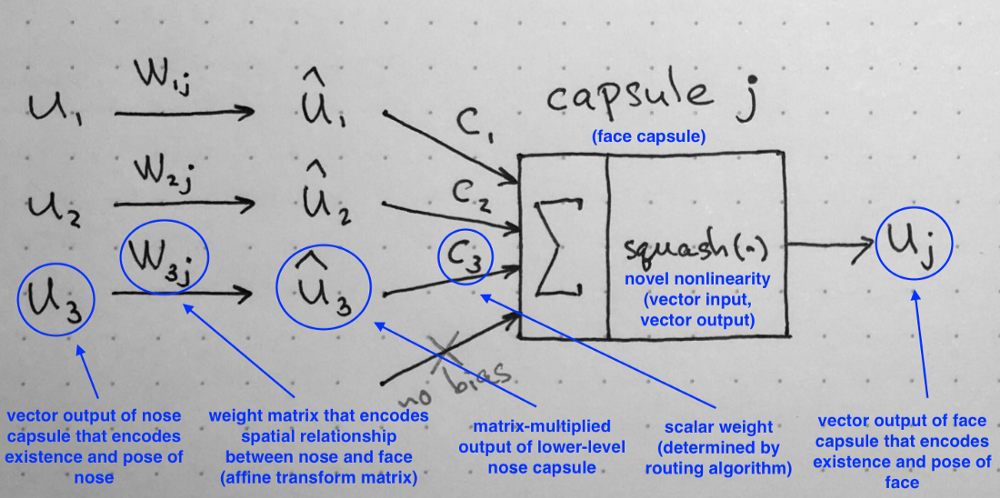
正如我在第二篇文章中所展示的那样，较下层的胶囊i需要决定如何将其输出矢量发送到更高级别的胶囊j。它通过改变标量权重c\_ij来做出这个决定，这个标量权重将会乘以它的输出矢量，然后作为输入，输入到一个更高级别的胶囊中。 符号方面，c\_ij表示将来自较下层胶囊i的输出矢量作为较高级别胶囊j的输入的权重。

关于权重c\_ij需要知道的事情：

1. 每个权重都是非负标量
2. 所有较低级别的胶囊i的权重总和等于1
3. 对于较低级别胶囊i，权重个数等于更高级别胶囊的个数
4. 这些权重由迭代动态路由算法确定

The first two facts allow us to interpret weights in probabilistic terms. Recall that the length a capsule’s output vector is interpreted as probability of existence of the feature that this capsule has been trained to detect. Orientation of the output vector is the parametrized state of the feature. So, in a sense, for each lower level capsule *i*, its weights *c\_ij* define a probability distribution of its output belonging to each higher level capsule *j*.

前两个事实允许我们用概率术语解释权重。 回想一下胶囊的输出向量的长度被解释为这个胶囊已经被训练检测的特征的存在概率。 输出矢量的方向是特征的参数化状态。 所以，从某种意义上说，对于每个较低级别的胶囊，其权重c\_ij定义了属于每个较高级别的胶囊j的输出的概率分布。



Recall: computations inside of a capsule as described in Part II of the series.

回想本系列文章的第二部分所描述的胶囊内部的计算。

**Dynamic Routing Between Capsules**

So, what exactly happens during dynamic routing? Let’s have a look at the description of the algorithm as published in the paper. But before we dive into the algorithm step by step, I want you to keep in your mind the main intuition behind the algorithm:

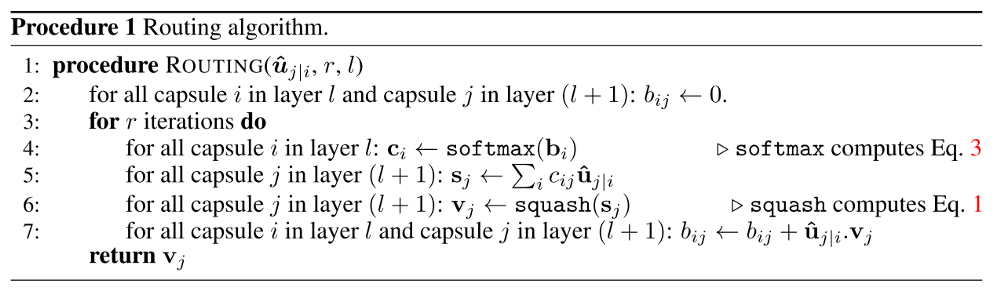
那么，在动态路由中究竟发生了什么呢？ 我们来看看本文发表的算法的描述。 但是在我们逐步深入研究算法之前，我希望您能够牢记算法背后的主要直觉：

*Lower level capsule will send its input to the higher level capsule that “agrees” with its input. This is the essence of the dynamic routing algorithm.*

*较低级别胶囊会将其输入发送到与其输入“一致”的较高级别胶囊。 这是动态路由算法的本质。*

Now that we have this in mind, let’s go through the algorithm line by line.

现在我们有了这个想法，让我们一行一行地看看算法。



Dynamic routing algorithm, as published in the [original paper](https://arxiv.org/abs/1710.09829).

The first line says that this procedure takes all capsules in a lower level *l* and their outputs *u\_hat*, as well as the number of routing iterations *r*. The very last line tells you that the algorithm will produce the output of a higher level capsule *v\_j*. Essentially, this algorithm tells us how to calculate [forward pass](http://cs231n.github.io/optimization-2/) of the network.

第一行表示这个过程将所有的胶囊都放在较低的级别l和他们的输出u\_hat，以及路由迭代次数r。 最后一行告诉你，算法将产生一个更高级别胶囊v\_j的输出。 本质上，这个算法告诉我们如何计算网络的正向传递。

In the second line you will notice that there is a new coefficient *b\_ij* that we haven’t seen before. This coefficient is simply a temporary value that will be iteratively updated and, after the procedure is over, its value will be stored in *c\_ij*. At start of training the value of *b\_ij* is initialized at zero.

在第二行，你会注意到有一个新的系数b\_ij，我们以前没有见过。 这个系数只是一个临时值，将被迭代更新，并且在过程结束后，它的值将被存储在c\_ij中。 训练开始时，b\_ij的值初始化为零。

Line 3 says that the steps in 4–7 will be repeated *r* times (the number of routing iterations).

第3行表示4-7中的步骤将被重复r次（路由迭代次数）。

Step in line 4 calculates the value of vector *c\_i* which is all routing weights for a lower level capsule *i*. This is done for all lower level capsules. Why [softmax](https://en.wikipedia.org/wiki/Softmax_function" \t "_blank)? Softmax will make sure that each weight *c\_ij* is a non-negative number and their sum equals to one. Essentially, softmax enforces probabilistic nature of coefficients *c\_ij* that I described above.

第4行中的步骤计算矢量c\_i的值，它是所有较低级别胶囊i的路由权重。 这是为所有较低级别的胶囊完成的。 为什么选择softmax？ Softmax将确保每个权重c\_ij是一个非负数，它们的和等于1。 本质上，softmax强化了我上面描述的系数c\_ij的概率性质。

At the first iteration, the value of all coefficients *c\_ij* will be equal, because on line two all *b\_ij* are set to zero. For example, if we have 3 lower level capsules and 2 higher level capsules, then all *c\_ij* will be equal to 0.5. The state of all *c\_ij* being equal at initialization of the algorithm represents the state of maximum confusion and uncertainty: lower level capsules have no idea which higher level capsules will best fit their output. Of course, as the process is repeated these uniform distributions will change.

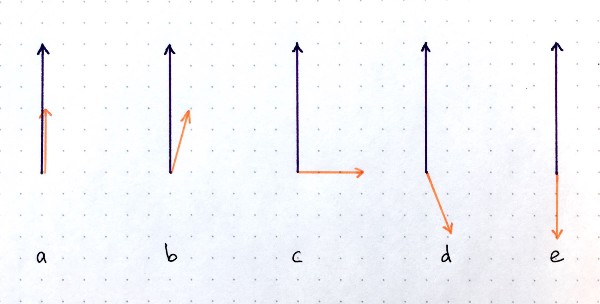
在第一次迭代中，所有系数c\_ij的值将是相等的，因为在第二行上所有的b\_ij被设置为零。 例如，如果我们有3个低级别胶囊和2个高级别胶囊，则所有c\_ij将等于0.5。 在算法初始化时，所有c\_ij的状态都是相等的，表示最大的困惑和不确定性状态：较低级别的胶囊不知道哪个较高级别的胶囊最适合它们的输出。 当然，随着这个过程的重复，这些均匀分布将会改变。

After all weights *c\_ij* were calculated for all lower level capsules, we can move on to line 5, where we look at higher level capsules. This step calculates a linear combination of input vectors, weighted by routing coefficients *c\_ij*, determined in the previous step. Intuitively, this means scaling down input vectors and adding them together, which produces output vector *s\_j*. This is done for all higher level capsules.

在对所有较低级别的胶囊计算了所有的权重c\_ij之后，我们可以移动到第5行，在那里我们查看更高级别的胶囊。 这个步骤计算输入向量的线性组合，由前一步确定的路由系数c\_ij加权。 直观地说，这意味着缩小输入矢量并将它们加在一起，这产生输出矢量s\_j。 这是为所有更高级别的胶囊完成的。

Next, in line 6 vectors from last step are passed through the squash nonlinearity, that makes sure the direction of the vector is preserved, but its length is enforced to be no more than 1. This step produces the output vector *v\_j* for all higher level capsules.

接下来，在第6行中，来自最后一步的矢量经过了南北非线性，这确保了矢量的方向被保留，但是其长度被强制为不大于1.该步骤产生所有更高级别的输出矢量v\_j胶囊。



[Dot product](https://en.wikipedia.org/wiki/Dot_product) is an operation that takes 2 vectors and outputs a scalar. There are several scenarios possible for the two vectors of given lengths but different relative orientations: (a) largest positive possible values; (b) positive dot product; (c) zero dot product; (d) negative dot product; (e) largest possible negative dot product. You can think of the dot product as a measure of similarity in the context of CapsNets. Source: author.

点积是一个需要2个向量并输出标量的操作。 对于给定长度的两个向量，但是具有不同的相对方向，有几种情况可能：（a）最大可能的正值; （b）积极的点积; （c）零点产品; （d）负点产品; （e）最大可能的负点产品。 您可以将点积视为CapsNets上下文中相似度的度量。 来源：作者。

To summarize what we have so far: steps 4–6 simply calculate the output of higher level capsules. Step on line 7 is where the weight update happens. This step captures the essence of the routing algorithm. This steps looks at each higher level capsule *j* and then examines each input and updates the corresponding weight *b\_ij*according to the formula. The formula says that the new weight value equals to the old value plus the dot product of current output of capsule *j* and the input to this capsule from a lower level capsule *i*. The dot product looks at similarity between input to the capsule and output from the capsule. Also, remember from above, the lower level capsule will sent its output to the higher level capsule whose output is similar. This similarity is captured by the dot product. After this step, the algorithm starts over from step 3 and repeats the process *r* times.

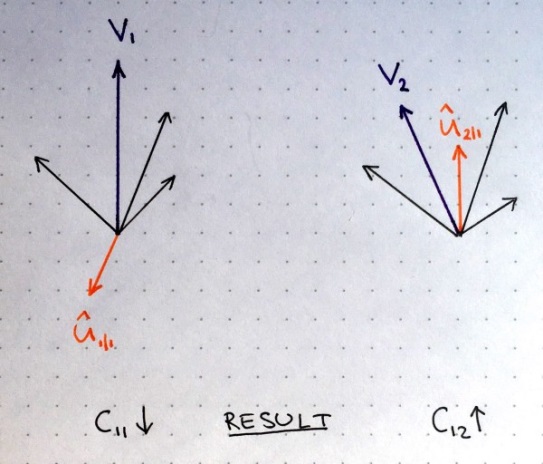
总结到目前为止：步骤4-6只是计算更高级别胶囊的输出。 第7行是更新重量的地方。 这一步捕获了路由算法的本质。 此步骤查看每个较高级别胶囊j，然后检查每个输入并根据公式更新相应的权重b\_ij。 公式说明新权重值等于旧值加上胶囊j的当前输出的点积和从较低级别胶囊i输入到该胶囊的输入。 点积看着胶囊的输入和胶囊的输出之间的相似性。 另外，从上面记得，低级别胶囊会将其输出发送到输出类似的高级别胶囊。 这种相似性被点积所捕获。 在此步骤之后，算法从步骤3开始重复该过程r次。

After *r* times, all outputs for higher level capsules were calculated and routing weights have been established. The forward pass can continue to the next level of network.

在r次之后，计算更高级别胶囊的所有输出并且建立路由权重。 正向传球可以延续到下一级的传球。

**Intuitive Example of Weight Update Step**

**权重更新步骤的直观示例**



In the figure on the left, imagine that there are two higher level capsules, their output is represented by purple vectors *v1* and *v2* calculated as described in previous section. The orange vector represents input from one of the lower level capsules and the black vectors represent all the remaining inputs from other lower level capsules.

在左边的图中，假设有两个更高级别的胶囊，它们的输出由紫色矢量v1和v2表示，如前一节所述。 橙色矢量表示来自其中一个较低级别胶囊的输入，而黑色矢量表示来自其他较低级别胶囊的所有其余输入。

We see that in the left part the purple output *v1* and the orange input *u\_hat* point in the opposite directions. In other words, they are not similar. This means their dot product will be a negative number and as result routing coefficient *c\_11* will decrease. In the right part, the purple output *v2* and the orange input v\_hat point in the same direction. They are similar. Therefore, the routing coefficient *c\_12* will increase. This procedure is repeated for all higher level capsules and for all inputs of each capsule. The result of this is a set of routing coefficients that best matches outputs from lower level capsules with outputs of higher level capsules.

我们看到在左边部分的紫色输出v1和橙色输入u\_hat指向相反的方向。 换句话说，它们并不相似。 这意味着它们的点积将是一个负数，结果路由系数c\_11会减少。 在右边部分，紫色输出v2和橙色输入v\_hat指向相同的方向。 他们是相似的。 因此，路由系数c\_12会增加。 对于所有更高级别的胶囊和每个胶囊的所有输入重复该过程。 这个结果是一组路由系数，它们与来自较低级别胶囊的输出与较高级别胶囊的输出最匹配。

**How Many Routing Iterations to Use?**

The paper examined a range of values for both MNIST and CIFAR data sets. Author’s conclusion is two-fold:

该文件检查了MNIST和CIFAR数据集的一系列值。 作者的结论是双重的：

1. More iterations tends to overfit the data
2. 更多的迭代往往会使数据过度拟合
3. It is recommended to use 3 routing iterations in practice
4. 建议在实践中使用3个路由迭代

Conclusion

In this article, I explained the dynamic routing algorithm by agreement that allows to train the CapsNet. The most important idea is that similarity between input and output is measured as dot product between input and output of a capsule and then routing coefficient is updated correspondingly. Best practice is to use 3 routing iterations.

在这篇文章中，我通过协议解释了动态路由算法，该协议允许训练CapsNet。 最重要的思想是，输入和输出之间的相似性被测量为胶囊的输入和输出之间的点积，然后相应地更新路由系数。 最佳做法是使用3个路由迭代。

In the next post, I will walk you through CapsNet architecture, where we will put together all pieces of the puzzle that we learned so far.

在下一篇文章中，我将带领你通过CapsNet体系结构，在这里我们将把迄今为止我们学到的所有难题集中到一起。