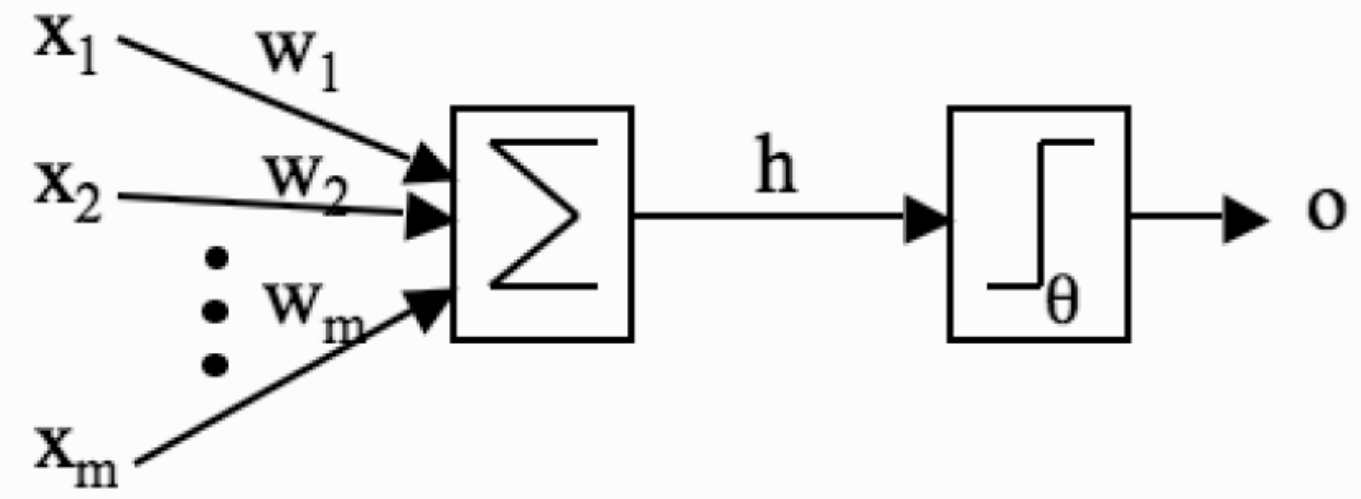
22 | 自适应的基函数：神经网络

2018-07-24 王天一

讲述：王天一
时长 18:44 大小 8.58M

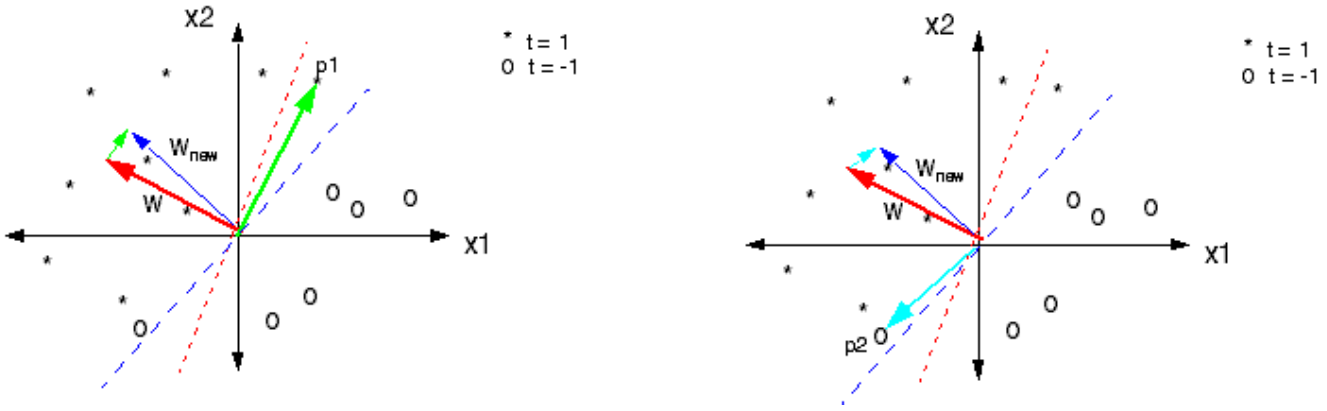
回眸人工神经网络的前半生，不由得让人唏嘘造化弄人。出道即巅峰的它经历了短暂的辉煌之后便以惊人的速度陨落，几乎沦落到人人喊打的境地。可谁曾想三十年河东三十年河西，一位天才的出现让神经网络起死回生，众人的态度也迅速从避之不及变成趋之若鹜。如果人工神经网络果真有一天如人所愿实现了智能，不知它会对自己的命运作何评价。

人工神经网络（artificial neural network）是对生物神经网络的模拟，意在通过结构的复制实现功能的复制。但人类神经系统在百万年进化中留下的智能密码并没有那么容易破解，因而神经网络最终也难以跳出统计模型的窠臼，成为**线性模型**大家族的又一位成员。



感知器示意图（图片来自 Machine Learning: an Algorithmic Perspective, 图 3.1）

人工神经网络的祖师爷是**感知器**（perceptron），其作用是根据输入数据的属性对它进行二分类。当偏置 $b = 0$ 时，感知器计算输入属性的线性组合 $w_1 x_1 + \cdots + w_n x_n$，所有参数 w_i 共通构成分类边界的法向量 $\mathbf{w}$。求出的线性组合接下来被送入**激活函数**（activation function）中计算结果。感知器的激活函数是**符号函数**，其输出的二元结果就表示了两种不同的类别。



感知器的学习过程示意图

（图片来自 https://www.willamette.edu/~gorr/classes/cs449/Classification/perceptron.html）

上图给出了感知器的学习过程。训练数据集是个线性可分的数据集，数据点的星号和圆圈代表不同的类别。感知器的初始参数是随机生成的，用这组随机参数生成的分类边界是图中的红色虚线。显然，在分类边界两侧的两个类别中都有误分类的点。

直观来看，要让走错片场的星号和圆圈找到组织，唯一的办法就是调整分类边界，让新边界把原始边界上方不同颜色的点区分开来。

调整的方法一目了然：既然星号集中在左侧而圆圈集中在右侧，那就要让分类边界向右侧旋转，把右侧的星号包进来，把左侧的圆圈踢出去。右侧子图表示的就是将原始边界试探性地旋转一个角度所得的结果。虽然这个小角度的旋转还是没能完全正确分类，却对分类的准确率有所改善。只要在此基础上进一步旋转，新的分类边界就可以将两个类别的点完全分开了。

由于分类时无需使用数据的概率分布，因此**感知器是一种基于判别式的分类模型**。但它和前面提到的线性判别分析又有所不同，其算法不是利用所有数据的统计特性一鼓作气计算出最优的参数，而是通过不断试错为参数优化过程提供反馈，从而实现动态的参数调整。**具备自适应的学习能力是感知器和前面所有模型相比独有的优势。**

在感知器的动态学习过程中，作为优化目标出现的是**感知准则**（perceptron criterion）。之所以没有选择常见的误分类率作为指标是因为它并不适用于参数的动态学习过程。

在分类时，产生每一种分类结果的分类边界都不是唯一的，这就让误分类率变成了关于参数 $\mathbf{w}$ 的分段常数函数。这不仅会使关于 $\mathbf{w}$ 的误分类率存在间断点，在求解梯度时也无法给出关于参数移动方向的信息。

感知准则虽然也是建立在误分类率的基础上，但它完全回避了上面的那些缺点，其表达式可以写成

$$E_P(\mathbf{w}) = -\sum_{x \in X_M} \mathbf{w}^T x_n t_n$$

其中 X_M 是由所有误分类点组成的集合，这说明分类正确的点都具有零误差。感知准则的基本思路是让每个误分类点的贡献 $-\mathbf{w}^T x_n t_n$ 都大于 0，这就保证了感知准则整体上的非负性。

二元变量 t_n 可以看成是数据点的真实类别和预测类别的差值，其作用在于控制每个误分类点的贡献。如果一个正类被判定为负类，那 t_n 就是个大于 0 的值，可以取成 +1；反过来负类被判定成正类时，t_n 则取 -1。当感知准则取得最小值 0 时，所有的数据点都被正确分类，感知器算法也就完全收敛。

对上面定义的感知准则求解梯度，可以得到每个轮次中参数的更新方式，也就是

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta x_n t_n$$

其中的超参数 η 是学习率（learning rate），τ 表示的是算法的轮次。这个表达式是感知器算法的**批处理更新原则**（batch update principle）。

根据这个算法的角度回头看上面的示意图，可以获得更清晰的解释。左侧子图中的 p1 点是个被误判为负类的正类点，其系数 $t_n = +1$。

要让这个点被正确分类，原始的系数向量 w 就要和向量 p1 与学习率的乘积相加，其几何意义就是向 p1 接近，移动的结果就是图中的 w_{new}。

位于第三象限的 p2 同样是误分类点，但是是被误判为正类。当负类被误判为正类时，t_n 的取值为 -1，此时原始的系数向量 w 要和向量 p2 与学习率的乘积相减，这里的减法体现为右侧子图中两个天蓝色箭头的方向区别。相减的几何意义是让新系数 w_{new} 远离误分类点 p2，不难看出，它和上面对 p1 的操作具有相同的效果。

感知器模型可以进一步推广为**多层感知器**（multilayer perceptron），也就是神经网络。最简单的神经网络是多个感知器的线性集成，神经网络的总输出是对每个感知器单独输出的线性组合进行非线性变换。

放在线性模型的大框架下，具有单个隐藏层的神经网络的数学表达式可以写成

$$f(x) = \sigma[\sum_{j=1}^{M} \beta_j h(\sum_{i=1}^{N} \alpha_i x_i + \alpha_0) + \beta_0]$$

其中 $\sigma(\cdot)$ 是输出层的激活函数，其最常见的选择是**对数几率函数**，这时输出层实质上就是个逻辑回归分类器。

除了对数几率函数之外，**双曲正切函数**$tanh$（hyperbolic tangent）也是不错的选择，$tanh$ 的值域是 [-1, +1]，这让它的特性和对数几率函数略有差别。

$h(\cdot)$ 表示的是隐藏层的激活函数，它既可以与 $\sigma(\cdot)$ 相同，也可以选取其他的非线性函数。α_i 和 β_j 分别表示了隐藏层和输出层的权重系数与偏置。

从上面的公式中不难看出，**神经网络的每个神经元都可以看成是做了基函数扩展的线性模型**：非线性的激活函数不仅将输出变成了输入属性的非线性函数，也变成了权重系数的非线性函数，体现的是整体的非线性处理。

当所有的激活函数都取恒等函数时，神经网络就将退化成最基础的线性回归。但神经网络的动态学习能力可以自适应地调整模型的参数，也就是线性组合中的权重系数。

神经网络的一个创新之处在于隐藏层的引入。除了输入层和输出层之外，所有无法直接观察的层都属于隐藏层（hidden layer）。**隐藏层的输出可以看成是某种导出特征（derived feature），它并不直接存在于输入之中，却可以根据对输入特征的挖掘推导出来。**神经网络的强大之处就是能够自适应地提取并修正人造特征，从而适配到数据中潜在存在的模式，深度学习优异的性能便由此而来。

在解决分类问题时，对神经网络的参数优化依赖于对**交叉熵**（cross-entropy）的最小化。网络输出的分类结果 t 满足两点分布，它关于数据 $\mathbf{x}$ 和参数 $\mathbf{w}$ 的似然概率可以写成

$$p(t|\mathbf{x},\mathbf{w})=y(\mathbf{x},\mathbf{w})^t[1-y(\mathbf{x},\mathbf{w})]^{1-t}$$

其中 $y(\mathbf{x},\mathbf{w})$ 是输出层激活函数为对数几率函数时的输出，可以视为 $\mathbf{x}$ 归属于正类的条件概率。求解上面式子的负对数似然，得到的就是交叉熵误差函数

$$E(\mathbf{w})=-\sum_{n=1}^{N}[t_n\ln y_n+(1-t_n)\ln(1-y_n)]$$

交叉熵的最小化与似然概率的最大化是等效的。误差函数的最小值可以通过反向传播（backpropagation）方法来求解，这在上一季的专栏中已经有过介绍，这里就不重复了。

神经网络中隐藏神经元的数目决定着网络的泛化性能，足够多的神经元可以实现任意复杂的函数，却也会带来严重的过拟合倾向，因而通过正则化的手段来控制网络的复杂度和性能是非常必要的。

一种简单的策略是**权重衰减**（weight decay），它与前面介绍过的岭回归类似，也是通过添加二次的正则化项来避免过拟合。权重衰减的误差函数可以写成

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

这里的 λ 是个超参数，控制着权重衰减的幅度。

另一种经常应用于神经网络中的正则化是**早停**（early stopping）。早停是建立在验证数据集上的正则化策略，简单地说就是对每一轮次训练出的模型计算出其在验证集上的性能，并将当前模型的参数和超参数存储下来。

在之后的每一轮训练中，训练结果在验证集上的性能都被拿来和先前存储的模型性能进行比较，之后保留两者中表现较好的模型的配置。这种策略和感知器训练中的口袋算法（pocket algorithm）类似。

如果从贝叶斯的角度去分析神经网络，模型训练的任务就变成了根据先验假设和训练数据来计算未知参数的后验分布，再对参数的分布积分来计算预测结果，写成数学表达式就是

$$p(y^{new}|x^{new}, D) = \int p(y^{new}|x^{new}, \mathbf{w})p(\mathbf{w}|D)\mathrm{d}\mathbf{w}$$

其中的 D 表示数据集。积分式中的第一项在分类中就对应着对数几率函数的输出，第二项则是参数的后验概率。对神经网络的贝叶斯分析涉及大量的复杂运算，所以我在这里就只介绍一些基本的思路。

在用于分类的神经网络中，先验假设就是参数 $\mathbf{w}$ 的概率分布，这个分布通常被处理成**零均值的高斯分布**。这个高斯分布的参数 α 又可以用超先验来表示。由于分类结果是离散的随机变量，它不像连续变量一样存在估计值和真实值的偏差，也就不用对这部分误差定义先验了。

虽然参数本身的先验分布是高斯形式，但激活函数的非线性特性会导致给定数据时参数对于数据的后验概率不满足高斯分布，这时就需要使用**拉普拉斯近似**（Laplace approximation）生成一个高斯分布，作为对未知后验的模拟。拉普拉斯近似的具体细节在这里暂且略过，你只需要知道它的用处就可以了。

在生成高斯分布时，拉普拉斯近似需要找到后验概率的一个最大值，这等效于对添加正则化项的误差函数 $\tilde{E}(\mathbf{w})$ 进行最小化，其中的正则化项就是参数 $\mathbf{w}$ 先验分布的体现。利用复杂的数值计算方法可以求出后验概率的最大值，进而确定后验概率的高斯近似。

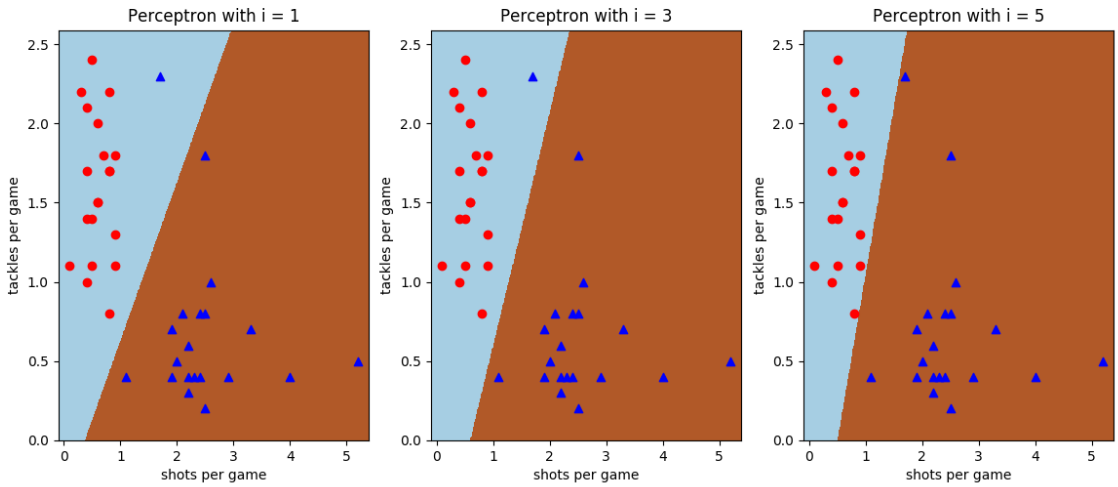
处理完了参数 $\mathbf{w}$，还要处理超参数 α。在后验概率的计算中，和参数 $\mathbf{w}$ 相关的超参数被看成是已知的固定值。但在计算预测结果时，还需要考虑超参数 α 的分布。

对参数 $\mathbf{w}$ 进行积分可以得到数据关于超参数的似然分布，也就是**边际似然函数**（marginal likelihood）。对边际似然函数进行最优化可以得到关于超参数 α 的点估计。由于非线性的激活函数让积分难以计算，通常会假设参数的后验概率非常窄，再利用使后验概率最大的参数来计算未知数据的输出。

神经网络是非参数模型的一种，它利用激活函数对线性模型做出了非线性的扩展，让每个输出变成了权重系数的非线性函数，从而在整体上拟合出非线性的效果。更重要的是，它引入了隐藏神经元与隐藏层，这种特殊的结构能够对原始的特征实现重构，从而给数据分析带来了更多的可能。

在 Scikit-learn 中，实现感知器的类 Perceptron 属于线性模型模块 linear_model，这样设置的原因无疑在于感知器本身属于线性判别模型。由于前面使用的中锋 - 中卫数据集是个线性可分的数据集，因此可以用感知器来进行分类。

下图给出了感知器对数据集的分类结果，从左到右的迭代次数分别为 1，3 和 5。可以看出，当初始分类结果有误时，感知器的算法会不断将分类边界向误分类点的方向调整，直到分类正确为止。



感知器对中锋 - 中卫数据集的分类结果

之前使用过的另一个分类数据集——曼城 - 西布朗数据集是个线性不可分的数据集，可以用它来验证多层感知器的性能。实现多层感知器的类叫做 MLPClassifier，在神经网络模块

neural_network 之中。但由于这个数据集是近似线性可分的，即使使用多层感知器也不会生成明显的非线性判决边界，你可以自己运行一下代码并观察结果。

今天我和你分享了感知器和神经网络的基本原理，包含以下四个要点：

神经网络是一类非线性模型，利用非线性的激活函数对输入的线性组合进行分类；

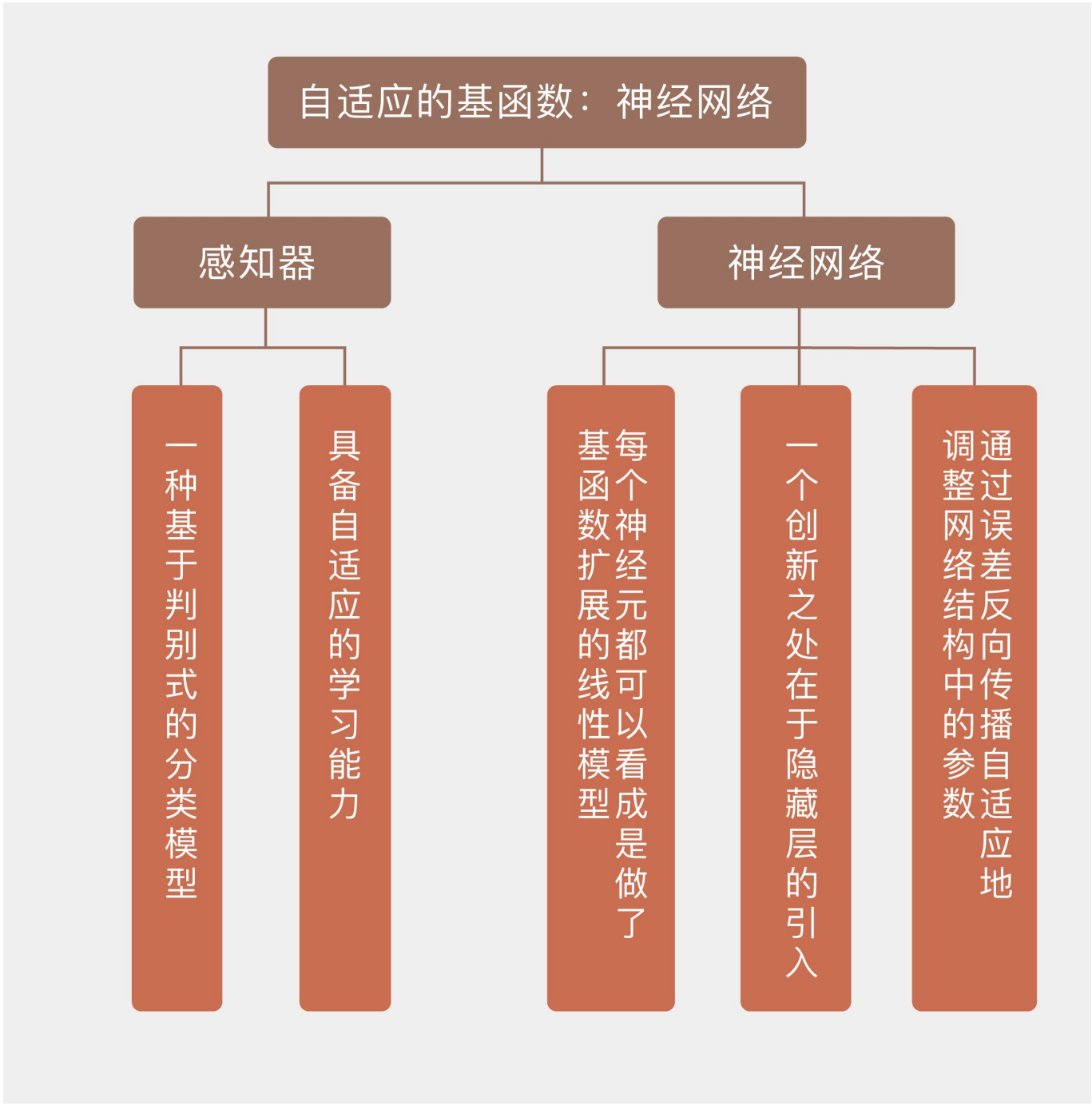
神经网络可以通过误差反向传播自适应地调整网络结构中的参数；

神经网络中隐藏层的作用是构造出新的导出特征；

用贝叶斯方法分析神经网络时，需要使用近似方法来应对非线性导致的计算问题。

2017 年时，神经网络的元老宿耆乔弗雷·辛顿（Geoffrey Hinton）提出了“胶囊网络”（capsule）的概念。胶囊由神经网络单个层中的若干神经元组合而成，可以看成是层中的一个子层。胶囊可以执行大量的内部计算，并输出一个矢量形式的结果，获得更多的输出信息。

你可以查阅关于胶囊网络的资料，思考它对神经网络的发展有何意义，并在这里分享你的见解。


自适应的基函数：神经网络
感知器
神经网络
一种基于判别式的分类模型
具备自适应的学习能力
每个神经元都可以看成是做了基函数扩展的线性模型
一个创新之处在于隐藏层的引入
通过误差反向传播自适应地调整网络结构中的参数

上一篇　21 | 基函数扩展：属性的非线性化

下一篇　23 | 层次化的神经网络：深度学习

精选留言 (7)

写留言

杨森　3

2018-07-30

我虽然难以理解清晰，但是这种方式我是喜欢的，已经不再是基础课了，目的是打通任督二脉。必须到一定难度才能触及到本质。一篇文章看一天我觉得不为过，不懂的名词需要慢慢查询学习了解。希望能度过难关，柳暗花明。

展开

作者回复: 是的，不能一直都在基础课转悠。如果看不懂细节可以先理解思想方法，搞清楚模型的思想原理，融会贯通之后再来深入细节。

吴常亮
2018-07-25

1

越来越看不懂了，还是需要老师深入浅出的讲解下去

展开

吕胜
2018-07-25

1

麻烦老师增加一些直观的解释

展开

风的轨迹
2019-03-14

老师我想问一个问题:
“在用于分类的神经网络中，先验假设就是参数 ww 的概率分布，这个分布通常被处理成零均值的高斯分布。”
确实我们在进行神经网络的参数初始化的时候一般都是用高斯分布，但是为什么必须用高斯分布而不能用别的分布呢?

展开

Python
2019-01-27

WC，看到今天终于懂老师写的专栏文章牛逼的地方了，之前看的懵懵懂懂，现在算有点前后融汇的感觉了

zhoujie
2018-09-05

一位天才的出现这里天才是指的谁？Hilton吗?

展开

作者回复: 是的

zhoujie
2018-09-05

关于深度学习，除了那本圣经之外还有其他书籍推荐吗

展开

作者回复: Francois Collet的Deep Learning with Python据说不错，但我没读过