麦子学院 www.maizieduc.com 百万级Python导师亲身指导 保你120天变身Python大牛 有时候，你需要的只是一句点拨

从头开始实现神经网络：入门

2015/09/21 • 系列教程 • 11 评论 • 神经网络

分享到： 20 本文由 伯乐在线 - fzr 翻译，唐尤华 校稿。未经许可，禁止转载！
英文出处：Denny Britz。欢迎加入翻译组。

获取代码：接下来，为了匹配文章的内容，所有的代码都会在Github上以iPython笔记的形式提供。

本文中我们会从头实现一个简单的3层神经网络。我们不会推导所有的数学公式，但会给我们正在做的事情一个相对直观的解释。我也会给出你研读所需的资源链接。

这里假设你已经比较熟悉微积分和机器学习的概念了。比如，你知道什么是分类和正则化。当然你也应该了解一点优化技巧，如梯度下降是如何工作的。但是即使你对上面提到的任何一个概念都不熟悉，你仍然会发现本文的有趣所在。

但是为什么要从头实现一个神经网络呢？即使你打算将来使用像PyBrain这样的神经网络库，从头实现神经网络仍然是一次非常有价值的练习。它会帮助你理解神经网络的工作原理，而这是设计有效模型的必备技能。

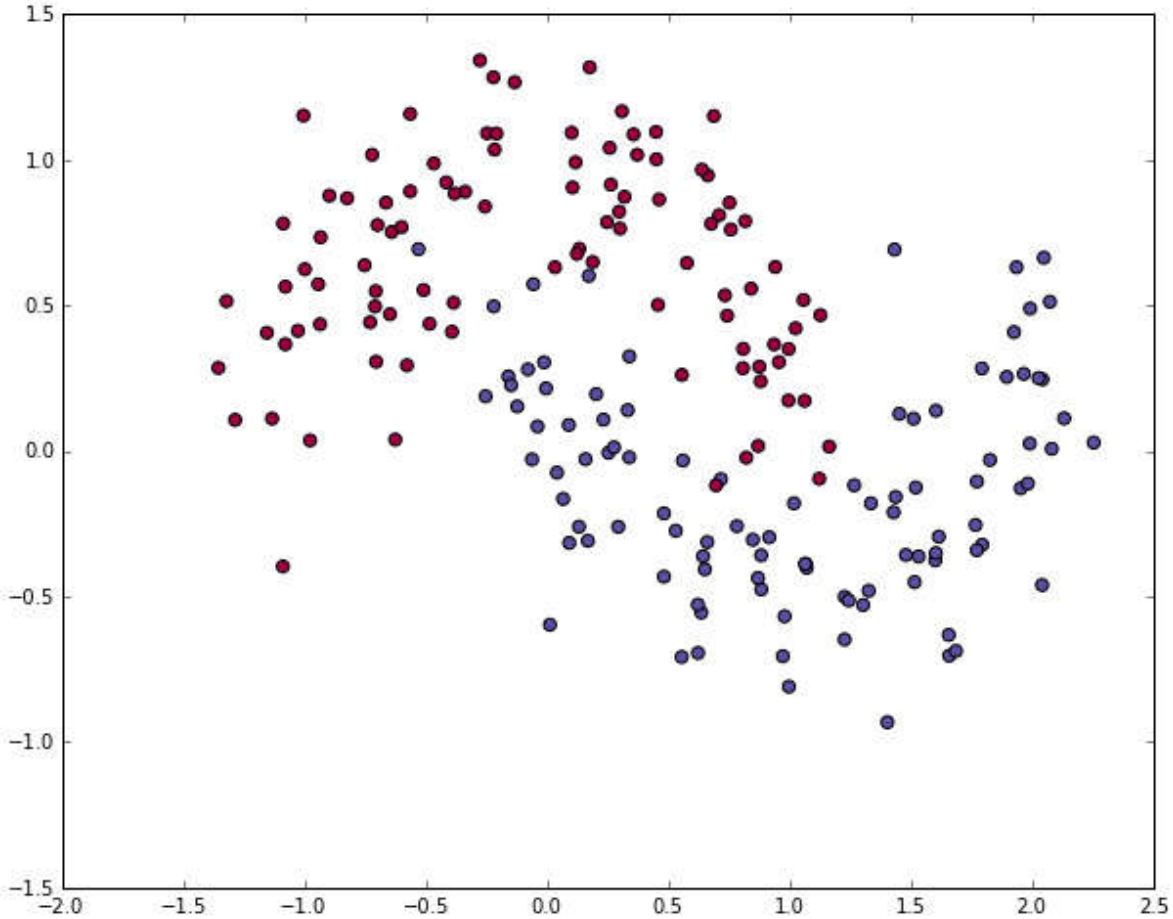
需要注意的是这里的示例代码并不是十分高效，它们本就是用来帮助理解的。在接下来的文章中，我会探索如何使用Theano写一个高效的神经网络实现。

产生数据集

让我们从力所能及的产生数据集开始吧。幸运的是，scikit-learn提供了一些很有用的数据集产生器，所以我们不需要自己写代码了。我们将从make_moons 函数开始。

Python

```python
# Generate a dataset and plot it
np.random.seed(0)
X, y = sklearn.datasets.make_moons(200, noise=0.20)
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)
```



产生的数据集中有两类数据，分别以红点和蓝点表示。你可以把蓝点看作是男性病人，红点看作是女性病人，x和y轴表示药物治疗。

我们的目标是，在给定x和y轴的情况下训练机器学习分类器以预测正确的分类（男女分类）。注意，数据并不是线性可分的，我们不能直接画一条直线以区分这两类数据。这意味着线性分类器，比如Logistic回归，将不适用于这个数据集，除非手动构建在给定数据集表现很好的非线性特征（比如多项式）。

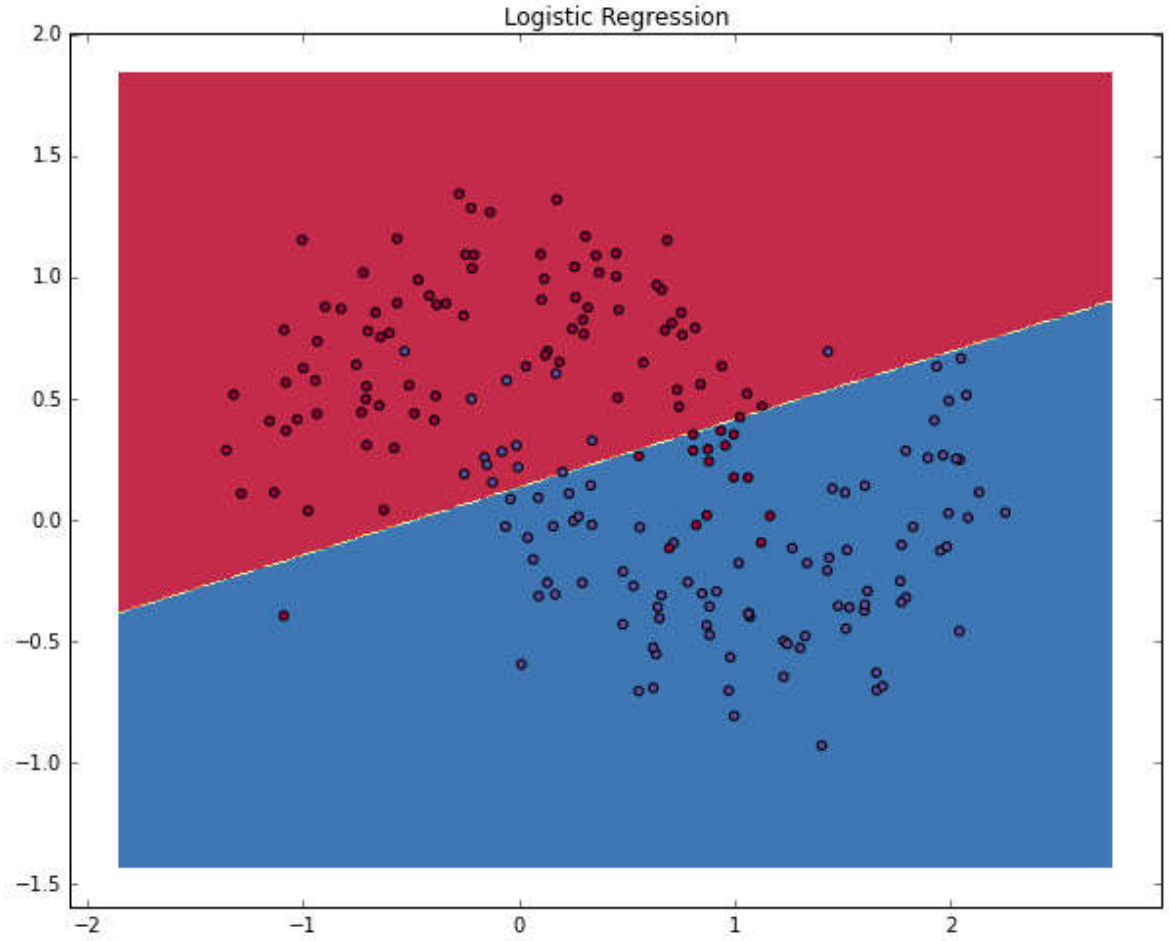
事实上，这也是神经网络的主要优势。你不用担心特征构建，神经网络的隐藏层会为你学习特征。

Logistic回归

为了证明这个观点，我们来训练一个Logistic回归分类器。它的输入是x和y轴的值，输出预测的分类（0或1）。为了简单，我们使用scikit-learn库里的Logistic回归类。

Python

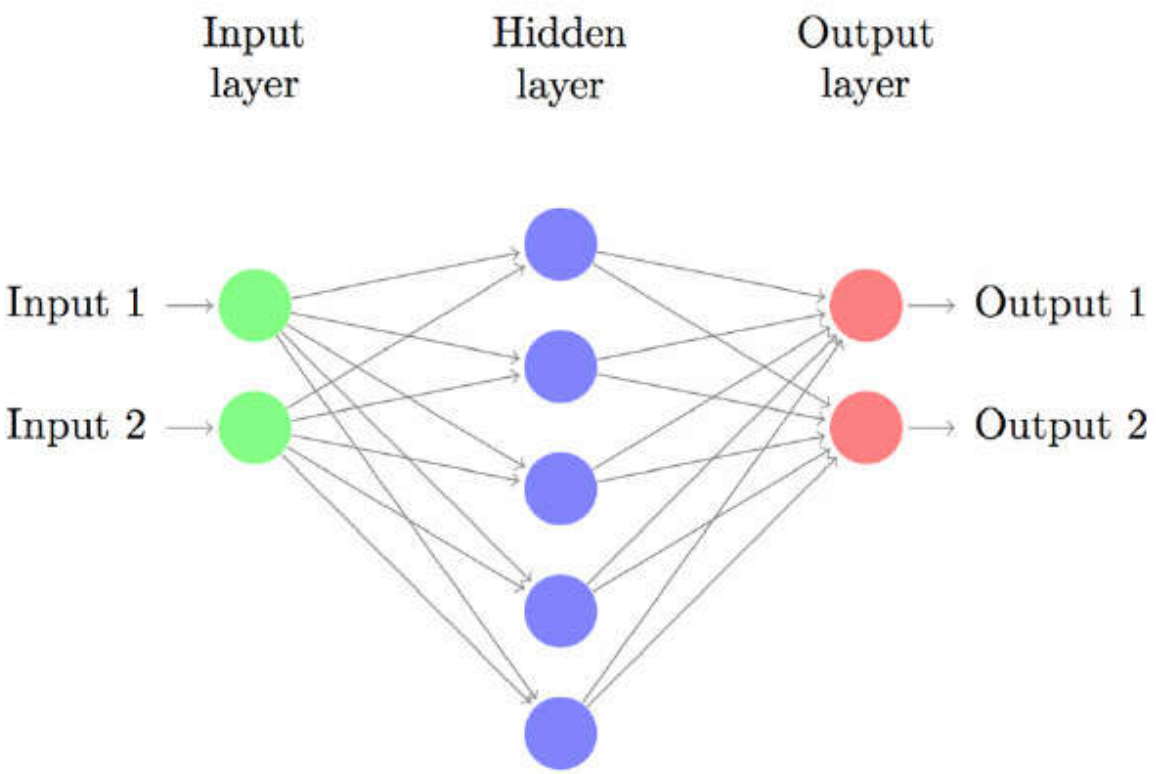
```python
# Train the logistic rgeression classifier
clf = sklearn.linear_model.LogisticRegressionCV()
clf.fit(X, y)

# Plot the decision boundary
plot_decision_boundary(lambda x: clf.predict(x))
plt.title("Logistic Regression")
```



上图展示了Logistic回归分类器学习到的决策边界。使用一条直线尽量将数据分离开来，但它并不能捕捉到数据的“月形”特征。

训练神经网络

让我们来建立具有一个输入层、一个隐藏层、一个输出层的三层神经网络。输入层的结点数由数据维度决定，这里是2维。类似地，输出层的结点数由类别数决定，也是2。（因为我们只有两类输出，实际中我们会避免只使用一个输出结点预测0和1，而是使用两个输出结点以使网络以后能很容易地扩展到更多类别）。网络的输入是x和y坐标，输出是概率，一个是0（女性）的概率，一个是1（男性）的概率。它看起来像下面这样：



我们可以为隐藏层选择维度（结点数）。放入隐藏层的结点越多，我们能训练的函数就越复杂。但是维度过高也是有代价的。首先，预测和学习网络的参数就需要更多的计算。参数越多就意味着我们可能会过度拟合数据。

如何选择隐藏层的规模？尽管有一些通用的指导和建议，但还是依赖于具体问题具体分析，与其说它是一门科学不如说是一门艺术。我们稍后会在隐藏层的结点数上多做一点事情，然后看看它会对输出有什么影响。

我们还需要为隐藏层挑选一个激活函数。激活函数将该层的输入转换为输出。一个非线性激活函数允许我们拟合非线性假设。常用的激活函数有tanh、the sigmoid函数或者是ReLUs。这里我们选择使用在很多场景下都能表现很好的tanh函数。这些函数的一个优点是它们的导数可以使用原函数值计算出来。例如，tanh x的导数是1-tanh^2 x。这个特性是很有用的，它使得我们只需要计算一次tanh x值，之后只需要重复使用这个值就可以得到导数值。

因为我们想要得到神经网络输出概率，所以输出层的激活函数就要是softmax。这是一种将原始分数转换为概率的方法。如果你很熟悉logistic回归，可以把softmax看作是它在多类别上的一般化。

神经网络如何预测

神经网络使用前向传播进行预测。前向传播只不过是一堆矩阵相乘并使用我们上面定义的激活函数了。假如x是该网络的2维输入，我们将按如下计算预测值（也是二维的）：

$$
\begin{aligned}
z_1 &= xW_1 + b_1 \\
a_1 &= \tanh(z_1) \\
z_2 &= a_1 W_2 + b_2 \\
a_2 &= \hat{y} = \mathrm{softmax}(z_2)
\end{aligned}
$$

zi是输入层、ai是输出层。W1,b1,W2,b2是需要从训练数据中学习的网络参数。你可以把它们看作是神经网络各层之间数据转换矩阵。看着上文的矩阵相乘，我们可以计算出这些矩阵的维度。如果我们的隐藏层中使用500个结点，那么有$W_1 \in \mathbb{R}^{2\times 500}, b_1 \in \mathbb{R}^{500}, W_2 \in \mathbb{R}^{500\times 2}, b_2 \in \mathbb{R}^{2}$.

现在你明白了为什么增大隐藏层的规模会导致需要训练更多参数。

学习参数

学习该网络的参数意味着要找到使训练集上错误率最小化的参数(W1,b1,W2,b2)。但是如何定义错误率呢？我们把衡量错误率的函数叫做损失函数（loss function）。输出层为softmax时多会选择交叉熵损失（cross-entropy loss）。假如我们有N个训练例子和C个分类，那么预测值（hat{y}）相对真实标签值的损失就由下列公式给出：

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{n\in N}\sum_{i\in C} y_{n,i}\log\hat{y}_{n,i}$$

这个公式看起来很复杂，但实际上它所做的事情不过是把所有训练例子求和，然后加上预测值错误的损失。所以，hat{y}（预测值）距离 hat{y}（真实标签值）越远，损失值就越大。

要记住，我们的目标是找到能最小化损失函数的参数值。我们可以使用梯度下降方法找到最小值。我会实现梯度下降的一种最普通的版本，也叫做有固定学习速率的批量梯度下降法。诸如SGD（随机梯度下降）或minibatch梯度下降通常在实践中有更好的表现。所以，如果你是认真的，这些可能才是你的选择，最好还能逐步衰减学习率。

作为输入，梯度下降需要一个与参数相关的损失函数的梯度（导数矢量）：，$\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$。为了计算这些梯度，我们使用了著名的后向传播算法。这个算法是从输出计算梯度的一种很高效的方法。在这里我不会深入讲解后向传播如何工作，但是在网络上流传有很多很优秀的讲解（参见这里或是这里）。

应用后向传播公式我们发现以下内容（这点你要相信我）：

$$\delta_3 = y - \hat{y}$$
$$\delta_2 = (1-\tanh^2 z_2)\circ\delta_3 W_2^T$$
$$\frac{\partial L}{\partial W_2} = a_1^T\delta_3$$
$$\frac{\partial L}{\partial b_2} = \delta_3$$
$$\frac{\partial L}{\partial W_1} = x^T\delta 2$$
$$\frac{\partial L}{\partial b_1} = \delta 2$$

实现

现在我们要准备开始实现网络了。我们从定义梯度下降一些有用的变量和参数开始：

Python

```python
num_examples = len(X) # training set size
nn_input_dim = 2 # input layer dimensionality
nn_output_dim = 2 # output layer dimensionality

# Gradient descent parameters (I picked these by hand)
epsilon = 0.01 # learning rate for gradient descent
reg_lambda = 0.01 # regularization strength
```

首先要实现我们上面定义的损失函数。以此来衡量我们的模型工作得如何：

Python

```python
# Helper function to evaluate the total loss on the dataset
def calculate_loss(model):
```

```
    W1, b1, W2, b2 = model['W1'], model['b1'], model['W2'], model['b2']
    # Forward propagation to calculate our predictions
    z1 = X.dot(W1) + b1
    a1 = np.tanh(z1)
    z2 = a1.dot(W2) + b2
    exp_scores = np.exp(z2)
    probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)
    # Calculating the loss
    corect_logprobs = -np.log(probs[range(num_examples), y])
    data_loss = np.sum(corect_logprobs)
    # Add regulatization term to loss (optional)
    data_loss += reg_lambda/2 * (np.sum(np.square(W1)) + np.sum(np.square(W2)))
    return 1./num_examples * data_loss
```

还要实现一个辅助函数来计算网络的输出。它的工作就是传递前面定义的前向传播并返回概率最高的类别。

Python

```
# Helper function to predict an output (0 or 1)
def predict(model, x):
    W1, b1, W2, b2 = model['W1'], model['b1'], model['W2'], model['b2']
    # Forward propagation
    z1 = x.dot(W1) + b1
    a1 = np.tanh(z1)
    z2 = a1.dot(W2) + b2
    exp_scores = np.exp(z2)
    probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)
    return np.argmax(probs, axis=1)
```

最后是训练神经网络的函数。它使用上文中发现的后向传播导数实现批量梯度下降。

Python

```
# This function learns parameters for the neural network and returns the model.
# - nn_hdim: Number of nodes in the hidden layer
# - num_passes: Number of passes through the training data for gradient descent
# - print_loss: If True, print the loss every 1000 iterations
def build_model(nn_hdim, num_passes=20000, print_loss=False):

    # Initialize the parameters to random values. We need to learn these.
    np.random.seed(0)
    W1 = np.random.randn(nn_input_dim, nn_hdim) / np.sqrt(nn_input_dim)
    b1 = np.zeros((1, nn_hdim))
    W2 = np.random.randn(nn_hdim, nn_output_dim) / np.sqrt(nn_hdim)
    b2 = np.zeros((1, nn_output_dim))

    # This is what we return at the end
    model = {}

    # Gradient descent. For each batch...
    for i in xrange(0, num_passes):

        # Forward propagation
        z1 = X.dot(W1) + b1
        a1 = np.tanh(z1)
        z2 = a1.dot(W2) + b2
        exp_scores = np.exp(z2)
        probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)

        # Backpropagation
        delta3 = probs
        delta3[range(num_examples), y] -= 1
        dW2 = (a1.T).dot(delta3)
        db2 = np.sum(delta3, axis=0, keepdims=True)
        delta2 = delta3.dot(W2.T) * (1 - np.power(a1, 2))
        dW1 = np.dot(X.T, delta2)
        db1 = np.sum(delta2, axis=0)

```

```
        # Add regularization terms (b1 and b2 don't have regularization terms)
        dW2 += reg_lambda * W2
        dW1 += reg_lambda * W1

        # Gradient descent parameter update
        W1 += -epsilon * dW1
        b1 += -epsilon * db1
        W2 += -epsilon * dW2
        b2 += -epsilon * db2

        # Assign new parameters to the model
        model = { 'W1': W1, 'b1': b1, 'W2': W2, 'b2': b2}

        # Optionally print the loss.
        # This is expensive because it uses the whole dataset, so we don't want to do it
        if print_loss and i % 1000 == 0:
          print "Loss after iteration %i: %f" %(i, calculate_loss(model))

    return model
```

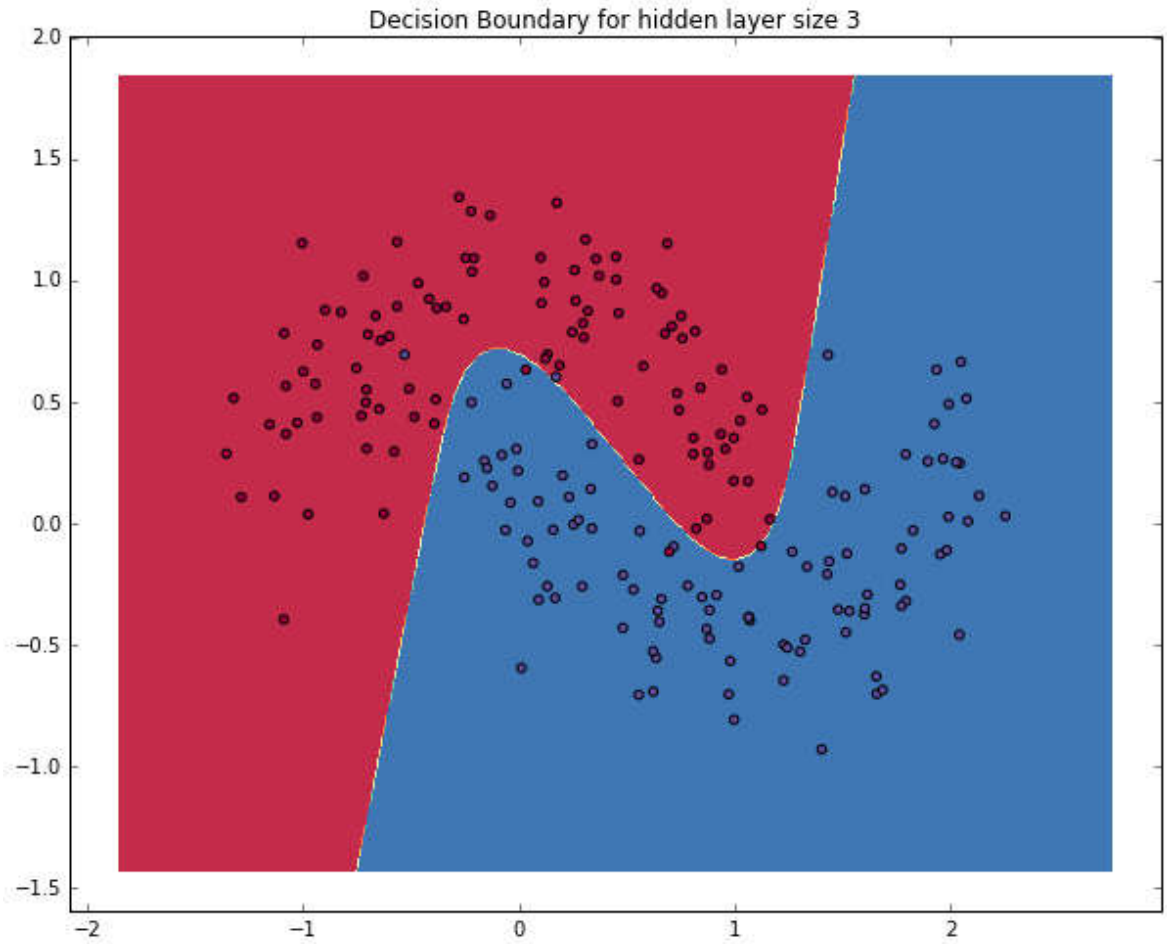
返回的model是一个dict

隐藏层规模为3的神经网络

一起来看看假如我们训练了一个隐藏层规模为3的神经网络会发生什么。

Python

```python
# Build a model with a 3-dimensional hidden layer
model = build_model(3, print_loss=True)

# Plot the decision boundary
plot_decision_boundary(lambda x: predict(model, x))
plt.title("Decision Boundary for hidden layer size 3")
```



耶！这看起来结果相当不错。我们的神经网络能够找到成功区分类别的决策边界。

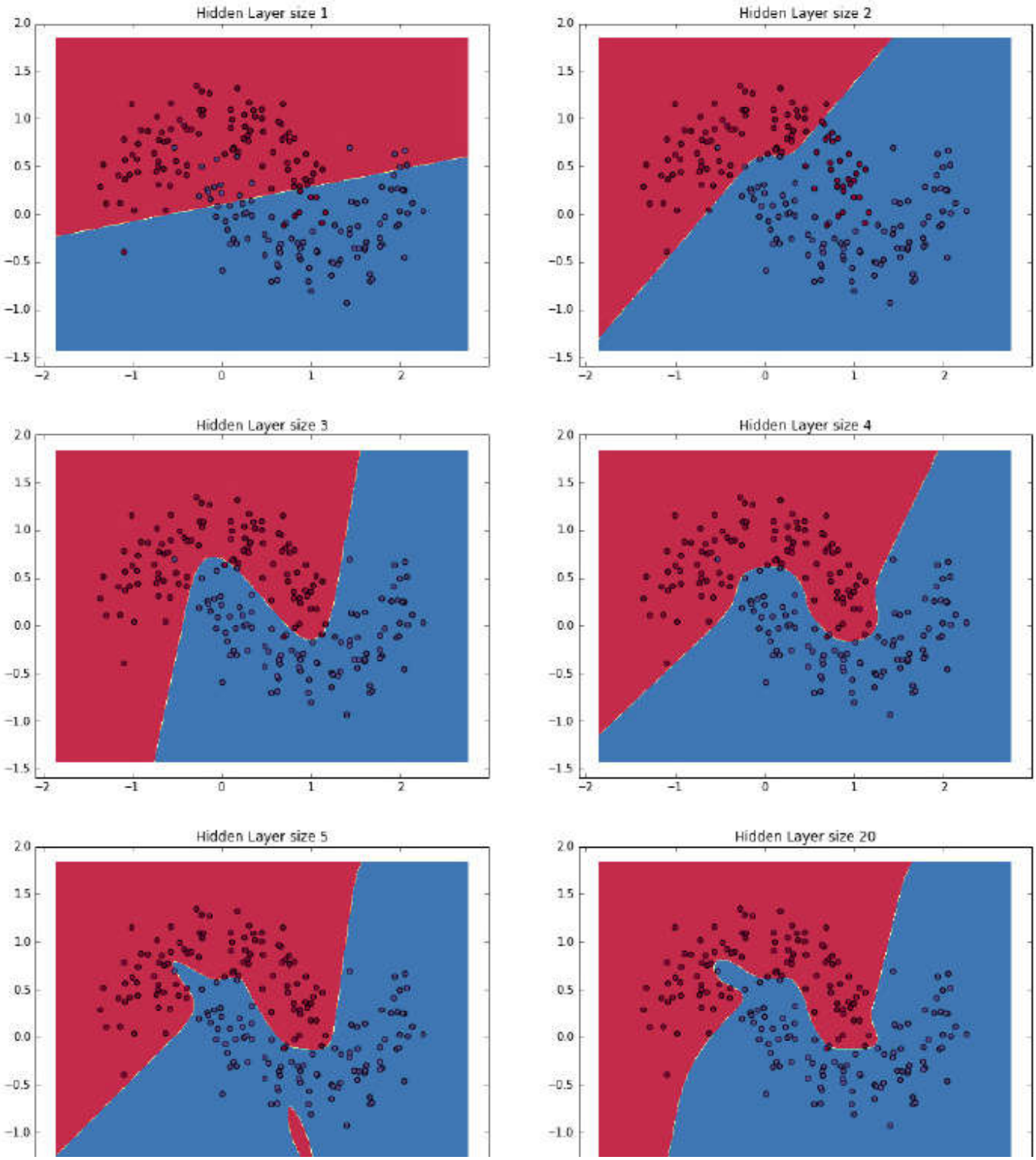
变换隐藏层的规模

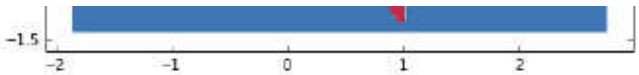
这里的神经网络还是3层，改变规模是指改变中间隐藏层节点的个数

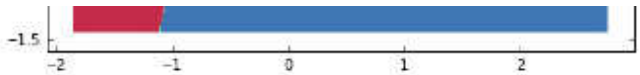
在上述例子中，我们选择了隐藏层规模为3。现在来看看改变隐藏层规模会对结果造成怎样的影响。

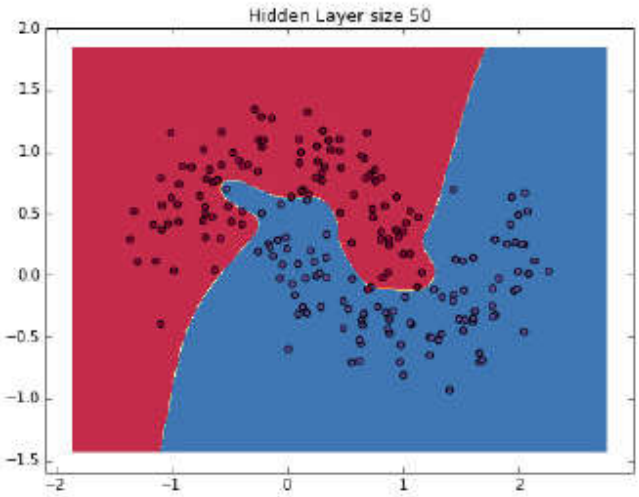
Python

```python
plt.figure(figsize=(16, 32))
hidden_layer_dimensions = [1, 2, 3, 4, 5, 20, 50]
for i, nn_hdim in enumerate(hidden_layer_dimensions):
    plt.subplot(5, 2, i+1)
    plt.title('Hidden Layer size %d' % nn_hdim)
    model = build_model(nn_hdim)
    plot_decision_boundary(lambda x: predict(model, x))
plt.show()
```









可以看到，低维隐藏层能够很好地捕捉到数据的总体趋势。更高的维度则更倾向于过拟合。它们更像是在“记忆”数据而不是拟合数据的大体形状。假如我们打算在独立测试集上评测该模型（你也应当这样做），隐藏层规模较小的模型会因为能更好的泛化而表现更好。虽然我们可以使用更强的泛化来抵消过拟合，但是为隐藏层选择一个合适的规模无疑是更加“经济”的方案。

练习

这里有一些练习帮助你进一步熟悉这些代码：

使用minibatch梯度下降代替批量梯度下降来训练网络（更多信息）。minibatch梯度下降在实际中一般都会表现的更好。

例子中我们使用的是固定学习速率的梯度下降ϵ。请为梯度下降的学习速率实现一个退火算法（更多信息）。

这里我们使用的是tanh$\tanh$ 作为隐藏层的激活函数。请尝试一下其他激活函数（比如上文中提到过的那些）。注意改变激活函数就意味着改变后向传播导数。

请扩展网络从两类输出到三类输出。你还需要为此产生一个近似数据集。

将这个网络扩展到四层。实验一下每层的规模。添加另一个隐藏层意味着前向传播和后向传播的代码都需要调整。

所有的代码都在Github上以iPython手册的形式提供。如有疑问或反馈，请留在评论中。

如果有人让你推荐Python技术书，请让他看这个列表 ->《经典Python技术书籍》

3 赞 18 收藏 11 评论

关于作者：fzr

微博：@fzr-fzr） 个人主页 • 我的文章 • 25

相关文章

- 人人都能用Python写出LSTM-RNN的代码！[你的神经网络学习最佳起步]
- 一个 11 行 Python 代码实现的神经网络
- 如何实现一个神经网络
- 机器学习算法原理之人工神经元和单层神经网络
- 如何用神经网络来恢复破损的密码？
- Python函数式编程指南（3）：迭代器
- 用 Python 从零开始写一个简单的解释器（2）
- Python函数式编程指南（4）：生成器
- Python 入门到精通(1)：Windows 搭建 Python 开发环境
- 动态规划用于解决重叠子问题的示例(Python版)

可能感兴趣的话题

- 从事开发几年了，开始遇到技术瓶颈了。怎么破？ • 3
- 有没有杭州的小伙伴们，可以周末组织个活动啥的 • 37
- 盛大2017校园招聘笔试题(C++)
- 网易2017校园招聘笔试题(哈希)
- 本人18了，学业无成。自学电脑现实吗？ • 42
- 如何保持单身的愉悦？ • 16
- 不加班的程序媛和程序猿们 • 72
- 大三学生前端自学和报班该选哪个 • 54
- 恒生电子2017校园招聘笔试题(数据类型) • 9
- 自学了一些前端基础知识，不知道接下来该怎么做 • 26

« 用 Python 从零开始写一个简单的解释器（1）
用 RAKE 和 Maui 做 NLP 关键词提取的教程 »

登录后评论 新用户注册

直接登录

最新评论

咔ァ佈奇の諾ｏｏＯ （ 1 ） 2015/09/22

请问这plot_decision_boundary函数是哪个包下的？？

赞 回复

Ky1XYZ （ 1 ） 2015/10/01

```
import matplotlib.pyplot as plt
import numpy as np
```

```
import sklearn
import sklearn.datasets
import sklearn.linear_model
import matplotlib

def plot_decision_boundary(pred_func):
# Set min and max values and give it some padding
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = 0.01
# Generate a grid of points with distance h between them
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
# Predict the function value for the whole gid
Z = pred_func(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Plot the contour and training examples
plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Spectral)

# Helper function to evaluate the total loss on the dataset
def calculate_loss(model):
W1, b1, W2, b2 = model['W1'], model['b1'], model['W2'], model['b2']
# Forward propagation to calculate our predictions
z1 = X.dot(W1) + b1
a1 = np.tanh(z1)
z2 = a1.dot(W2) + b2
exp_scores = np.exp(z2)
probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)
# Calculating the loss
corect_logprobs = -np.log(probs[range(num_examples), y])
data_loss = np.sum(corect_logprobs)
# Add regulatization term to loss (optional)
data_loss += reg_lambda/2 * (np.sum(np.square(W1)) + np.sum(np.square(W2)))
return 1./num_examples * data_loss

def predict(model, x):
W1, b1, W2, b2 = model['W1'], model['b1'], model['W2'], model['b2']
# Forward propagation
z1 = x.dot(W1) + b1
a1 = np.tanh(z1)
z2 = a1.dot(W2) + b2
exp_scores = np.exp(z2)
probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)
return np.argmax(probs, axis=1)

def build_model(nn_hdim, num_passes=20000, print_loss=False):

# Initialize the parameters to random values. We need to learn these.
np.random.seed(0)
W1 = np.random.randn(nn_input_dim, nn_hdim) / np.sqrt(nn_input_dim)
b1 = np.zeros((1, nn_hdim))
W2 = np.random.randn(nn_hdim, nn_output_dim) / np.sqrt(nn_hdim)
b2 = np.zeros((1, nn_output_dim))

# This is what we return at the end
model = {}

# Gradient descent. For each batch...
for i in xrange(0, num_passes):

# Forward propagation
z1 = X.dot(W1) + b1
a1 = np.tanh(z1)
z2 = a1.dot(W2) + b2
exp_scores = np.exp(z2)
probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)

# Backpropagation
delta3 = probs
delta3[range(num_examples), y] -= 1
dW2 = (a1.T).dot(delta3)
db2 = np.sum(delta3, axis=0, keepdims=True)
```

```
delta2 = delta3.dot(W2.T) * (1 - np.power(a1, 2))
dW1 = np.dot(X.T, delta2)
db1 = np.sum(delta2, axis=0)

# Add regularization terms (b1 and b2 don't have regularization terms)
dW2 += reg_lambda * W2
dW1 += reg_lambda * W1

# Gradient descent parameter update
W1 += -epsilon * dW1
b1 += -epsilon * db1
W2 += -epsilon * dW2
b2 += -epsilon * db2

# Assign new parameters to the model
model = { 'W1': W1, 'b1': b1, 'W2': W2, 'b2': b2}

# Optionally print the loss.
# This is expensive because it uses the whole dataset, so we don't want to do it too often.
if print_loss and i % 1000 == 0:
print "Loss after iteration %i: %f" %(i, calculate_loss(model))

return model

np.random.seed(0)
X, y = sklearn.datasets.make_moons(200, noise=0.20)
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)

clf = sklearn.linear_model.LogisticRegressionCV()
clf.fit(X, y)

num_examples = len(X) # training set size
nn_input_dim = 2 # input layer dimensionality
nn_output_dim = 2 # output layer dimensionality

# Gradient descent parameters (I picked these by hand)
epsilon = 0.01 # learning rate for gradient descent
reg_lambda = 0.01 # regularization strength

# Build a model with a 3-dimensional hidden layer
#model = build_model(3, print_loss=True)

# Plot the decision boundary
#plot_decision_boundary(lambda X: predict(model, X))
#plt.title("Decision Boundary for hidden layer size 3")
#plt.figure(figsize=(16, 32))
hidden_layer_dimensions = [1, 2, 3, 4, 5, 20, 50]
for i, nn_hdim in enumerate(hidden_layer_dimensions):
plt.subplot(5, 2, i+1)
plt.title('Hidden Layer size %d' % nn_hdim)
model = build_model(nn_hdim)
plot_decision_boundary(lambda X: predict(model, X))
plt.show()
```

赞 回复

咔ァ佈奇の諾oｏO （ 1 ） 2015/11/01

博主，请问您的目标函数是怎样的？

赞 回复

噗噗熊 2015/09/23

真的很不错

赞 回复

[redacted] （ 1 • ）

2015/09/26

产生数据集的代码不对，应该是

```
# Generate a dataset and plot it
np.random.seed(0)
X, y = sklearn.datasets.make_moons(200, noise=0.20)
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)
```

赞 回复

fzr （ 25 ）
开发/java

2015/09/26

是的，我的失误，不好意思，我会尽快修正的

赞 回复

黄利民 （ 88 • ）
站长

2015/09/26

1. 已代更新；
2. 你的权限也提升了；
3. 谢谢@[redacted] 提醒

赞 回复

fzr （ 25 ）
开发/java

2015/09/27

站长。。。能不说真名吗。。。。>_<

赞 回复

suichen （ 1 ）

2015/09/28

plot_decision_boundary函数怎么写的？？

赞 回复

KylXYZ （ 1 ）

2015/10/01

看了源码， 需要做一下小修改， 字母大小写 ‘x’ to ‘X’， 其余的都能编译， 多谢站长， 好文！
″ plot_decision_boundary(lambda x: predict(model, x)) ″->″ plot_decision_boundary(lambda X: predict(model, X)) ″

赞 回复

MyGuitarGentlyWeeps （ 1 • ）

2015/12/21

谁能告诉我，隐藏层挑选tanhx作为激活函数的依据是什么吗？如果层数再增加，每一层激活函数都不一样的话，这样计算不会变的更复杂吗？也许我的想法是错误的，想知道在哪里能够看到相关资料呢？

赞 回复

Python ▼
输入搜索关键字
搜索

Python小组话题 **我有新话题**

一个正则表达式的问题，请熟悉正则表...
苏宁莫 发起 • 2 回复

零基础自学Python感觉很难，不像大...
keepcalm 发起 • 55 回复

怎么感觉一直都没学会Python
艾凌风 发起 • 7 回复

Numpy模块安装问题
桃李不言 发起 • 2 回复

使用pymssql包的executemany出现value e...
wx_EFkn3LpN 发起

python3.6安装xlwt1.1.2报错
忽必列 发起 • 2 回复

- 本周热门Python文章
- 本月热门
- 热门标签

0 如果有人让你推荐Python技术书，请...

1 10天100小时学数据科学，我推荐你这...

2 Python 之父谈论 Python 的未来

3 爬取豆瓣电影top250提取电影分类进行...

4 Python要你命三千

5 爬虫学习之一个简单的网络爬虫

6 Python 代码调试技巧

7 爬虫学习之基于Scrapy的网络爬虫

8 Flask学习摘要

9 从源码编译 Python

Python工具资源 **更多资源 »**

python-recsys：一款实现推荐系统的python库
Python， 机器学习

YouCompleteMe：一款Vim代码补全引擎
C语言， Python， Vim， 文本编辑器扩展， 编辑器插件

Xlwt：Windows Office Excel表格生成库
Office

SimpleCV

SimpleCV：开源计算机视觉框架
Python， 计算机视觉

SciPy库：Python的科学计算工具集
科学计算和数据分析

最新评论

-
  Re: 如何用神经网络来恢复破损的密码？
  Hi，@蓝风铃 :我看到你在小组中发帖问 neupy 包如何下载。我们翻译的文章，开头就有英文链接...

- Re: 10天100小时学数据科学，我推荐...
  fantastic，this is my true love
- Re: Python 新手常犯错误（第二部...
  本质原因是python没有C++那种 int, float, 或是javascript 的var这...
- Re: 机器学习之用Python从零实现贝...
  是我搞错了，想复杂了。肤浅了
- Re: 如果有人让你推荐Python技术书...
  不错，先留着以后看看。
- Re: 机器学习之用Python从零实现贝...
  predict方法中，if bestLabel is None or probability >...
- Re: 如果有人让你推荐Python技术书...
  《Python核心编程》，当初看到书名以为是提高类的
- Re: 如何确定一个字符串中是否所有字...
  不过s最大256

关于 Python 频道

Python频道分享 Python 开发技术、相关的行业动态。

快速链接
问题反馈与求助 »
Python工具资源 »
Python技术话题 »

关注我们

新浪微博：@Python开发者
RSS：订阅地址
推荐微信号

Python开发者 Linux爱好者 数据库开发

合作联系
Email：bd@Jobbole.com
QQ： 2302462408 （加好友请注明来意）

更多频道

小组 - 好的话题、有启发的回复、值得信赖的圈子
头条 - 分享和发现有价值的内容与观点
相亲 - 为IT单身男女服务的征婚传播平台

资源 - 优秀的工具资源导航
翻译 - 翻译传播优秀的外文文章
文章 - 国内外的精选文章
设计 - UI,网页，交互和用户体验
iOS - 专注iOS技术分享
安卓 - 专注Android技术分享
前端 - JavaScript, HTML5, CSS
Java - 专注Java技术分享
Python - 专注Python技术分享

© 2016 伯乐在线 首页 博客 资源 小组 相亲 反馈