Learning of AI

冯迁

2017-2-1

目录

1 What's New

RNN: 时间递归神经网络 recurrent neural network，结构递归神经网络，recursive neural network 时间递归的神经元间连接构成有向图，而结构递归利用相似的神经网络结构递归构造更为复杂的深度网络。RNN 将状态在自身网络中循环传递，因此可以接受更广泛的时间序列结构输入。手写识别是最早成功利用 RNN 的研究结果 RNN 是某种体现出了随时间动态变化特性的神经网络。因此其在处理时间序列数据和过程上效果都很不错。这样的数据和过程正是语音识别和自然语言处理的常见研究对象。[2, 1]

Research at Google

Reurrent

1．1 编码器

将输入序列编码为一个隐藏状态, 隐藏状态方程可以是 Logistic 方程，也可以是 LSTM 单元。$p(y_t) = p(y_t \mid y_{t-1}, ..., y_1)$ 其中 y_t 是第 t 个位置上的输出，它的概率基于之前输出的所有词语。

以上概率通过隐藏状态来计算: $p(y_t) = g(y_{t-1}, \vec{h_t}, \vec{c})$ $\vec{c}$ 是所有隐藏状态的编码，总含了所有隐藏状态。这里的非线性方程 $g(y, h, c)$ 可以是一个复杂的前馈神经网络，也可以是简单的非线性方程（但有可能因此无法适应复杂的条件而得不到任何有用结果）。

1．2 解码器

1．3 双向读取

Recursive

结构递归神经网络是一类用结构递归的方式构建的网络，比如说递归自编码机（Recursive Autoencoder），在自然语言处理的神经网络分析方法中用于解析语句。

Boltzmann Machine,，迁移学习，

生成式模型:to recognize shapes,first learn to generate images-Geoffrey Hinton.

DBN:A summery of DNB

强化学习: 玩电子游戏 [3]，使用卷积网络来简化游戏界面的像素数据，将数据转化成一组特征的简化集合，最终这些信息被用来确定采用是样的操作。

音乐信息学:[4]

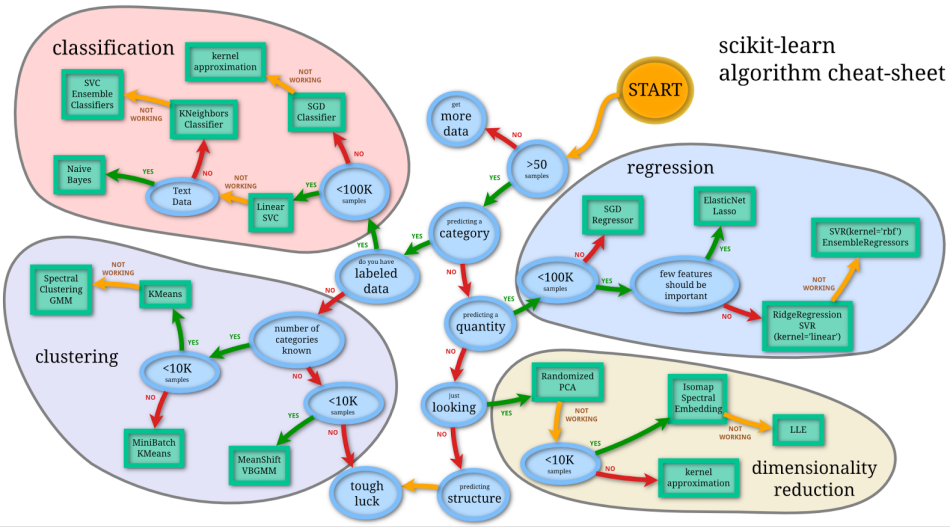
2 Classic

随机梯度下降,BP, 卷积网络, 正规化.

3 基础计划

3.1 Method Selection Suggestions

关于 sklearn 中方法的选择可参考:



经典算法之比较:

方法比较 注:Calibrated boosted trees 的性能最好，随机森林第二，uncalibrated bagged trees 第三,calibratedSVMs 第四，uncalibrated neural nets 第五。

性能较差的是朴素贝叶斯，决策树。有些算法在特定的数据集下表现较好. tensorflow-cookbook

3.2 PCA

参看skPCA

3.3 Autoencoder

3.4 KNN

参看skKNN

K 近邻：算法采用测量不同特征值之间的距离的方法进行分类。

优点：

1. 简单好用，容易理解，精度高，理论成熟，既可以用来做分类也可以用来做回归；

2. 可用于数值型数据和离散型数据；

3. 训练时间复杂度为 O(n)；无数据输入假定；

4. 对异常值不敏感

缺点：

1. 计算复杂性高；空间复杂性高；

2. 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；

3. 一般数值很大的时候不用这个，计算量太大。但是单个样本又不能太少否则容易发生误分。

4. 最大的缺点是无法给出数据的内在含义。

3.5 LinearRegression

参看skreg

3.6 LogisticRegression

3.7 SVM

代码参考SVM.py Svm

优点：

1. 可用于线性/非线性分类，也可以用于回归，泛化错误率低，计算开销不大，结果容易解释；

2. 可以解决小样本情况下的机器学习问题，可以解决高维问题可以避免神经网络结构选择和局部极小点问题。

3.SVM 是最好的现成的分类器，现成是指不加修改可直接使用。并且能够得到较低的错误率，SVM 可以对训练集之外的数据点做很好的分类决策。

缺点：对参数调节和和函数的选择敏感，原始分类器不加修改仅适用于处理二分类问题。

Logistic 回归：根据现有数据对分类边界线建立回归公式，依次进行分类。

优点：实现简单，易于理解和实现；计算代价不高，速度很快，存储资源低；

缺点：容易欠拟合，分类精度可能不高参看skSVM

利用超平面将数据分成两部分，从而完成对数据的分类. 基本上需要考虑的情况有线性的，非线性的，数据噪音情况的. 非线性情况解决的采用将数据映射到高维空间，再构造线性分类器，为降低计算复杂度，避免在映射后的高维空间中进行内积运算，可构造核函数来解决.

3.8 DecisionTrees

决策树

优点：

1. 概念简单，计算复杂度不高，可解释性强，输出结果易于理解；

2. 数据的准备工作简单，能够同时处理数据型和常规型属性，其他的技术往往要求数据属性的单一。

3. 对中间值得确实不敏感，比较适合处理有缺失属性值的样本，能够处理不相关的特征；

4. 应用范围广，可以对很多属性的数据集构造决策树，可扩展性强。决策树可以用于不熟悉的数据集合，并从中提取出一些列规则这一点强于 KNN。

缺点：

1. 容易出现过拟合；

2. 对于那些各类别样本数量不一致的数据，在决策树当中，信息增益的结果偏向于那些具有更多数值的特征。

3. 信息缺失时处理起来比较困难。忽略数据集中属性之间的相关性。首先参看skDecisionTrees 获得初步认识. 根据一些特征进行分类, 每个节点提出一个问题, 通过判断, 将数据分成两类, 再继续提问. 根据已有数据学习出问题, 再投入新数据的时候, 就可以根据这棵树上的问题, 将数据划分到合适的叶子上. 决策树 py 或者决策树 py 数据科学沉思录一个持续更新并且很不错的 io.

3.9 Random Forest

特点: 在当前所有算法中，具有极好的准确率; 能够有效地运行在大数据集上; 能够处理具有高维特征的输入样本，而且不需要降维；能够评估各个特征在分类问题上的重要性；在生成过程中，能够获取到内部生成误差的一种无偏估计；对于缺省值问题也能够获得很好得结果.

森林中任意两棵树的相关性：相关性越大，错误率越大；森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低.

减小特征选择个数 k，树的相关性和分类能力也会相应的降低；增大 k，两者也会随之增大。所以关键问题是如何选择最优的 k（或者是范围），这也是随机森林唯一的一个参数。

参看skForests 随机森林 py 随机森林是一个非常灵活的机器学习方法，从市场营销到医疗保险有着众多的应用。它可以用于市场营销对客户获取和存留建模或预测病人的疾病风险和易感性。随机森林能够用于分类和回归问题，可以处理大量特征，并能够帮助估计用于建模数据变量的重要性。

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法

随机森林可以用于几乎任何一种预测问题（包括非线性问题）。它是一个相对较新的机器学习策略（90 年代诞生于贝尔实验室）可以用在任何方面。它属于机器学习中的集成学习这一大类。

在原始数据中随机选取数据, 组成几个子集, 形成一系列决策树, 再对新数据进行分类; 换句话说: 由原始数据形成的矩阵 M, 从 M 中随机选取 K 个小矩阵, 小矩阵构成 k 个决策树, 将新数据投入到 k 个决策树中得到 k 个分类结果, 从中选取分类数目最大者为其类别.

3.10 Naive Bayes

朴素贝叶斯

优点：

1. 生成式模型，通过计算概率来进行分类，可以用来处理多分类问题，

2. 对小规模的数据表现很好，适合多分类任务，适合增量式训练，算法也比较简单。

缺点：

1. 对输入数据的表达形式很敏感，

2. 由于朴素贝叶斯的“朴素”特点，所以会带来一些准确率上的损失。

3. 需要计算先验概率，分类决策存在错误率。

3.11 Neural Network

3.12 monte carlo method

3.13 Markov Chains

3.14 MCMC

首先参考经典文献 [7]

3.15 Boltzmann Machine

3.16 GAN

初始主要学习来源:GAN. 代码学习主要根据 [5] 而来.

关于文中用到的分批规范化可以参看文章 [6],

内容来源GAN 知乎: 将 ML 算法分为两类: 生成式模型, 判别式模型; 其区别在于: 对于输入的 x, 类别标签 y, 在生成式模型中估计其联合概率分布, 而判别式模型估计其属于某类的条件概率分布. 常见的判别式模型包括: LogisticRegression,SVM,Neural Network..., 生成式模型包括:Naive Bayes,GMM,Bayesian Network,MRF...

收集集:GAN-paper-codes(Theano), 不同阶段的论文以及代码实现.
而 GAN,tf 的实现可参考GAN-Tensorflow.
最新进展GAN-news[8, 9, 10],以及代码实现 WGAN-torch. WGAN-Tensorflow.

3.17 杂乱的学习

tf-allbasic 无监督学习: 在人工神经网络中，自我组织映射和适应性共振理论也是常用的无监督学习，也有许多网络是利用无监督学习对数据进行特征提取，如在图像识别的应用中，可用无监督学习提取图片中物体的边缘特征。

目前许多有监督学习算法，如 SVM，DNN 或是 boosting，决策树等，都在工业界分类或决策任务上取得了不错的成果.

自编码器是一个数据降维压缩算法，算法训练的结果是拟合一个恒等函数 $h_{w,b}(x) = x$. 即通过自编码器的训练后，$decoder(encoder(x)) = x$. 目前自编码器主要有数据降噪和为数据可视化而降维两个方面的应用。

归一化
批量梯度下降 Batch gradient descent
随机梯度下降 Stochastic gradient descent
最小批梯度下降 Mini-batch gradient descent

前向计算和误差反向传播。

```
        # 前向计算实现：
# w:权值矩阵
# a:神经网络内部节点
# x:神经网络外部节点
def feedforward(w,a,x):
    # 激活函数 sigmoid
    f = lambda s: 1 / (1 + np.exp(-s))

   #将网络的内部及外部输入联合起来与权值矩阵进行加权叠加
    #这里使用的是矩阵运算使训练更加快捷
    w = np.array(w)
    temp = np.array(np.concatenate((a,x),axis=0))
    z_next = np.dot(w , temp)

    # 返回计算的下一层神经元的计算结果
    # 及未经过激活函数前的加权叠加结果
    return f(z_next), z_next
```

```
    #误差反向传播实现:
# w: 权值矩阵
# z: 当前层的未经过激活函数前的神经元值
# delta_next:  下一层的
def backprop(w,z,delta_next):

    # sigmoid 激活函数
    f = lambda s: np.array(1 / (1 + np.exp(-s)))

    # 激活函数 sigmoid 的导数
    df = lambda s: f(s) * (1 - f(s))

    # 误差反向传播计算上一层的    并返回
    delta = df(z) * np.dot(w.T,delta_next)

    return delta
```

自编码器的完整代码实现查看自编码器 或者本地 autoencoder 目录。建议先去推到其公式.

部分维基百科定义:

广义的人脸识别实际包括构建人脸识别系统的一系列相关技术，包括人脸图像采集、人脸定位、人脸识别预处理、身份确认以及身份查找等；而狭义的人脸识别特指通过人脸进行身份确认或者身份查找的技术或系统。

4 辅助工具

4.1 Pandas

4.2 numpy

数据结构重组 hstack(): 将第一维的进行并排 (横向), 等价于:np.concatenate(tup, axis=1)

vstack()(纵向), 等价于: np.concatenate(tup, axis=0) if tup contains arrays thatare at least 2-dimensional.

stack():axis=0,1,-1

dstack(): 等价于: np.concatenate(tup, axis=2)

vsplit():numpy.vsplit(ary, indices or sections) 相当于剪切

concatenate():numpy.concatenate((a1, a2, ...), axis=0)

从以上可以看到，只需熟练掌握 concatenate 即可.

4.3 sklearn

参考文献

[1] Zaremba, Wojciech. "An empirical exploration of recurrent network architectures." (2015).

[2] Zaremba, Wojciech, and Ilya Sutskever. "Learning to execute." arXiv preprint arXiv:1410.4615 (2014).

[3] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[4] Humphrey, Eric J., Juan P. Bello, and Yann LeCun. "Feature learning and deep architectures: new directions for music informatics." Journal of Intelligent Information Systems 41.3 (2013): 461-481.

[5] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[6] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[7] Andrieu, Christophe, et al. "An introduction to MCMC for machine learning." Machine learning 50.1-2 (2003): 5-43.

[8] Goodfellow, Ian. "NIPS 2016 Tutorial: Generative Adversarial Networks." arXiv preprint arXiv:1701.00160 (2016).

[9] Arjovsky, Martin, and Léon Bottou. "Towards principled methods for training generative adversarial networks." NIPS 2016 Workshop on Adversarial Training. In review for ICLR. Vol. 2016. 2017.

[10] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." arXiv preprint arXiv:1701.07875 (2017).