第十一章　深度学习

机器学习与 python 实践

Recommend
火龙果软件

深度学习

- 深度学习，一般指多层网络结构以及相应的训练方法，这些方法大部分都是基于神经网络。

  对比：浅层学习，我们熟悉的很多模型和算法例如 SVM，回归树和分类树以及树的集成等。
- 图像识别，语音识别，自然语言处理，人工智能
- 机器学习的主流方法、偏工程、算力要求
- 同时包含了非线性建模、特征提取和降维三个特点
- （深度）监督学习：常见的神经网络如 DNN、CNN、RNN
  （深度）无监督学习：VAE 和 GAN
  （深度）强化学习

Recommend
火龙果软件

目录

1. 前馈神经网络和反向传播算法
2. 网络结构
3. 自编码和生成模型
4. 揭开深度学习的黑箱?

Recommend
火龙果软件

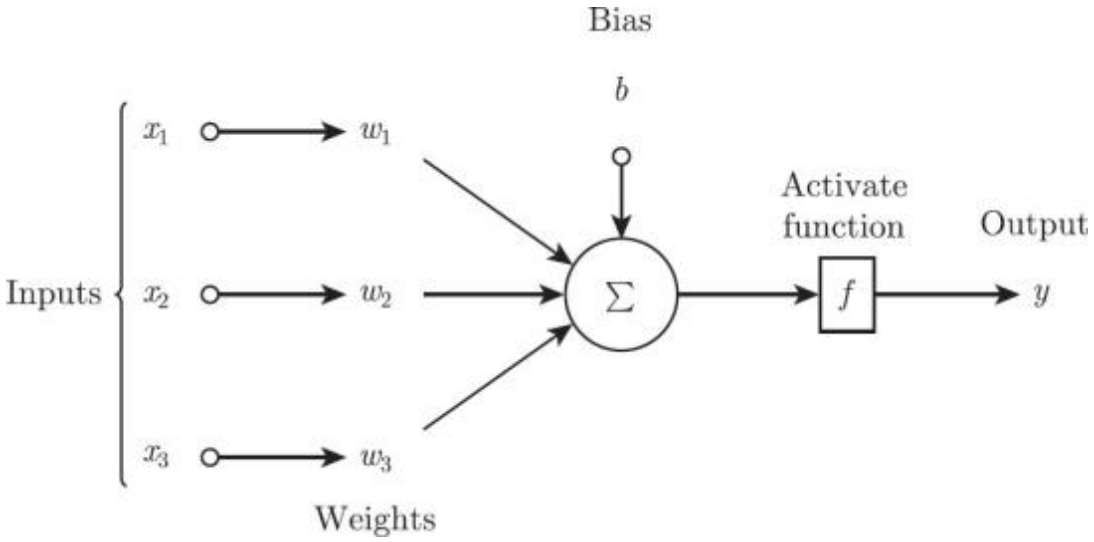
目录

1. 前馈神经网络和反向传播算法
2. 网络结构
3. 自编码和生成模型
4. 揭开深度学习的黑箱?

Recommend
火龙果软件

神经元

- 神经元：包含输入输出和信息处理的单元
  神经网络：由多个神经元组成。
- 感知器（Perception）：$y = f(z) = f(\mathbf{x}^T\mathbf{w} - b)$



$\mathbf{x}$：输入信息，$\mathbf{w}$：权重向量，$z = \mathbf{x}^T\mathbf{w}$：输入信息汇总，
f：激活函数，$y = f(z)$：经过激活函数处理产生的输出。
b：偏差项，类似回归的截距项。

Recommend 火龙果软件

神经元 (感知器)

- 感知器结构：

$$y = f(z) = f(\mathbf{x}^T\mathbf{w} - b)$$

- 权重的迭代更新规则：

$$\mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} + \eta(y_i - \hat{y}_i^{(l)})\mathbf{x}_i^T$$

$$\hat{y}_i^{(l+1)} = f(\mathbf{x}_i^T\mathbf{w}^{(l+1)})$$

其中 $\eta > 0$ 是学习率 (learning rate)。,

- 目标函数：最小化

$$-\sum_{i \in M} y_i(\mathbf{x}_i^T\mathbf{w})$$

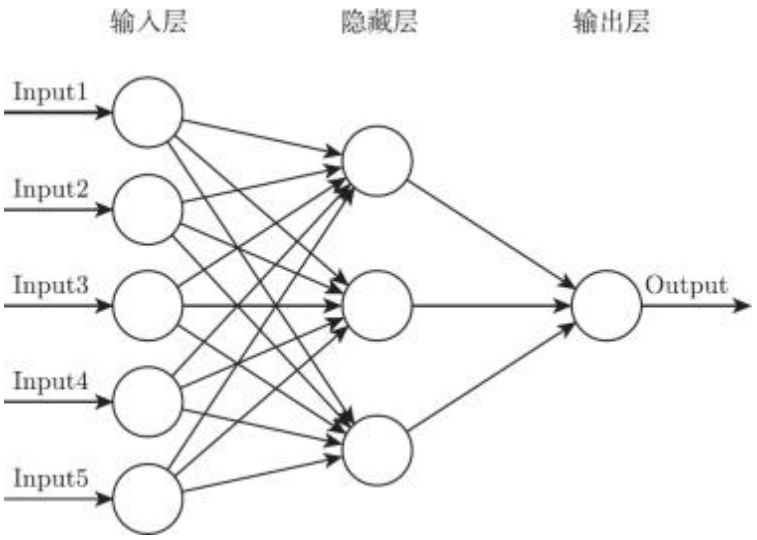
其中 M 是错误分类点的集合。

- 线性可分时：感知器算法可以在有限步内收敛到一个分割超平面。
- 线性不可分时：支持向量机可以看成是对感知器的改进和拓展，引入基扩张后可以有效处理。

Recommend 火龙果软件

前馈神经网络

- 单隐含层前馈神经网络



隐含层和输出层拥有激活函数。
输入层和输出层结点固定不变，隐含层的层数及神经元数可调节

- 深度神经网络：具有多个隐含层的神经网络
- 深度学习：不同类型的深度神经网络及其中权重的有效训练方法

Recommend 火龙果软件

梯度下降算法

- 神经网络训练中最常用的算法之一。
- 目标函数：最小化

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} L(y_i, f(x_i;\boldsymbol{\theta}))$$

L：损失函数，f：神经网络结构，$\boldsymbol{\theta}$：神经网络的权重参数

- 参数更新过程：

$$\begin{aligned}\boldsymbol{\theta}^{(l+1)} &= \boldsymbol{\theta}^{(l)} - \eta \frac{\partial \ell}{\partial \boldsymbol{\theta}} \\ &= \boldsymbol{\theta}^{(l)} - \eta \sum_{i=1}^{n} \frac{\partial L(y_i, f(x_i;\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}\end{aligned}$$

其中 η 为学习率（Learning rate）。

- 样本量很少的凸优化问题，可以很快达到全局最优。
如果数据量太大，参数更新一次需要大量计算。
- 改进：随机梯度下降算法（SGD）

Recommend 火龙果软件

梯度下降算法改进

- 随机梯度下降算法（SGD）：
  梯度下降算法的随机近似。每次迭代只使用一个训练数据更新参数

$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \eta \frac{\partial L(y_i, f(x_i;\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$$

  SGD 需要以某种顺序遍历训练样本多次，直到收敛。

- mini-batch SGD：将数据分多个小批量的子集，每次取出一个子集样本 S 来更新参数

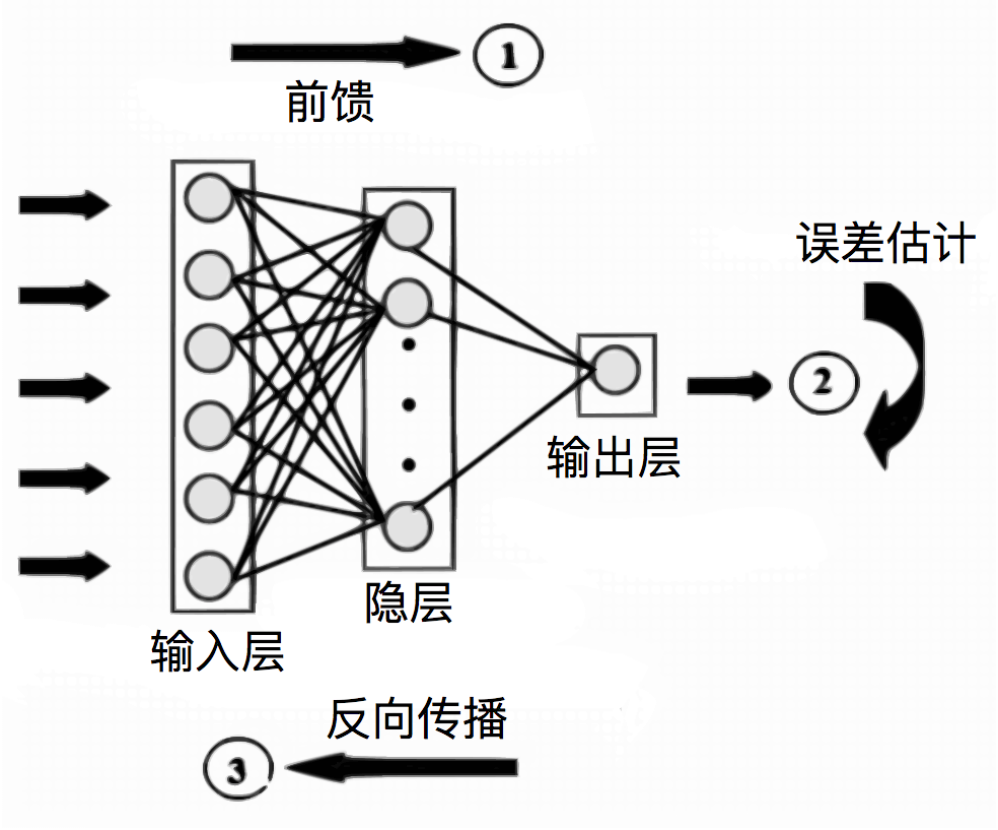
$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \eta \sum_{i \in S} \frac{\partial L(y_i, f(x_i;\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$$

  mini-batch 的大小一般设为 128 或者 256。

Recommend 火龙果软件

反向传播算法

- 反向传播 (BP)：实现梯度下降算法的有效途径



反向传播算法

- 反向传播 (BP)：实现梯度下降算法的有效途径

1. 最小化残差平方和

$$\ell(\theta) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{K} \beta_j \sigma(\mathbf{w}_j^T \mathbf{x}_i) \right)^2$$

2. 梯度下降法更新参数 θ

$$\beta_j^{(l+1)} = \beta_j^{(l)} - \eta \partial \ell / \partial \beta_j^{(l)}$$
$$w_{jk}^{(l+1)} = w_{jk}^{(l)} - \eta \partial \ell / \partial w_{jk}^{(l)}$$

其中

$$\partial \ell / \partial \beta_j^{(l)} = -2 \sum_{i=1}^{N} \{y_i - f^{(l)}(\mathbf{x}_i)\} \sigma(\mathbf{w}_j^{(l)T} \mathbf{x}_i)$$

$$\partial \ell / \partial w_{jk}^{(l)} = -2 \sum_{i=1}^{N} \{y_i - f^{(l)}(\mathbf{x}_i)\} \beta_j^{(l)} \sigma'(\mathbf{w}_j^{(l)T} \mathbf{x}_i) x_{ik}$$

反向传播算法

记 $\delta_i^{(l)} = y_i - f^{(l)}(\mathbf{x})_i$，有

$$\begin{aligned}
\partial \ell / \partial \beta_j^{(l)} &= -2 \sum_{i=1}^{N} \delta_i^{(l)} \sigma(\mathbf{w}_j^{(l)T} \mathbf{x}_i) \\
s_{ij}^{(l)} &= \delta_i^{(l)} \beta_j^{(l+1)} \sigma'(\mathbf{w}_j^{(l)T} \mathbf{x}_i) \\
\partial \ell / \partial w_{jk}^{(l)} &= -2 \sum_{i=1}^{N} s_{ij}^{(l)} x_{ik}
\end{aligned}$$

- 反向传播算法
  1. forward pass：信息从输入层逐层向前传播
     根据当前的参数 $\mathbf{w}_j^{(l)}$ 和 $\beta_j^{(l)}$，依次算出隐含层 $\sigma(\mathbf{w}_j^{(l)T} \mathbf{x}_i)$ 和残差 $\delta_i^{(l)} = y_i - f^{(l)}(\mathbf{x})_i$
  2. backward pass：误差从输出层逐层向后传播，
     根据以上三式，和梯度下降更新式，依次更新 β_j 和 $\mathbf{w}_j$。

随机梯度算法的改进

- Momentum：对梯度进行加权平均，以指数平均为例。

$$m^{(l)} = \beta m^{(l-1)} + (1-\beta)\nabla J^{(l)}$$

$$\theta^{(l+1)} = \theta^{(l)} - \eta\, m^{(l)}$$

- RMSProp：对梯度的平方加权指数平均，并更新学习率，

$$v^{(l)} = \beta v^{(l-1)} + (1-\beta)(\nabla J^{(l)})^2$$

$$\theta^{(l+1)} = \theta^{(l)} + \eta \frac{\nabla J^{(l)}}{\sqrt{v^{(l)} + \epsilon}}$$

- Adam：结合动量和 RMSProp，

$$m^{(l)} = \beta_1 m^{(l-1)} + (1-\beta_1)\nabla J^{(l)}$$

$$v^{(l)} = \beta_2 v^{(l-1)} + (1-\beta_2)(\nabla J^{(l)})^2$$

$$\hat{m}^{(l)} = m^{(l)}/(1-\beta_1)^l$$

$$\hat{v}^{(l)} = v^{(l)}/(1-\beta_2)^l$$

$$\theta^{(l+1)} = \theta^{(l)} - \eta \hat{m}^{(l)} / \sqrt{v^{(l)} + \epsilon}$$

激活函数和梯度消失问题

- 震荡（Zigzag）问题

  若激活函数恒大于零，使用随机梯度下降算法更 $\boldsymbol{\beta}$ 时，

$$\frac{\partial \ell}{\partial \beta_j^{(l)}} = -2\{y_i - f^{(l)}(\mathbf{x}_i)\} \quad \sigma(\mathbf{w}_j^{(l)T}\mathbf{x}_i)$$

  所有 β_j 梯度同号，梯度更新时可能会产生来回震荡，算法收敛慢

- 梯度消失问题

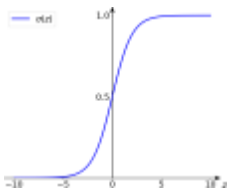
  导数在某区间之外取值很小时：

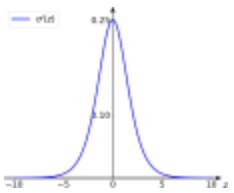
  使用反向传播算法时导数逐渐变为 0，网络参数很难被更新。
  某些神经元可能永远也不会被激活，参数无法更新，
  这种现象被称作“死区”，也是梯度消失的一种表现形式。

激活函数和梯度消失问题

- Sigmoid 函数 [震荡][梯度消失]

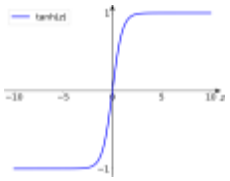
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

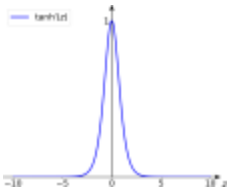




- tanh 函数 [梯度消失]

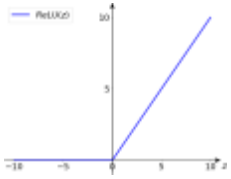
$$tanh(z) = \frac{2}{1+e^{-2z}} - 1$$

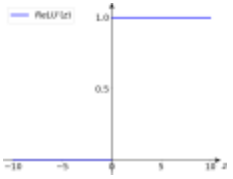




- 修正线性单元 (ReLU)[$z < 0$梯度消失]

$$ReLU(z) = \begin{cases} z & z > 0 \\ 0 & z \le 0 \end{cases}$$





- Leaky ReLU

$$\text{Leaky ReLU}(z) = \begin{cases} z & z > 0 \\ \alpha z & z \le 0 \end{cases}$$

- 参数化 ReLU(PReLU)
  随机化 ReLU(RReLU)
  ELU

案例分析

- 目标和背景：

  目标：预测未来一天股票的涨跌概率，并根据预测结果构建投资组合。

- 解决方案和程序：

  把逻辑回归方法替换为前馈神经网络，其他部分保持不变。
  可以看到，在预测效果和组合回报上，神经网络方法比逻辑回归有较大提升，但略差于 GBDT 和 XGBoost。

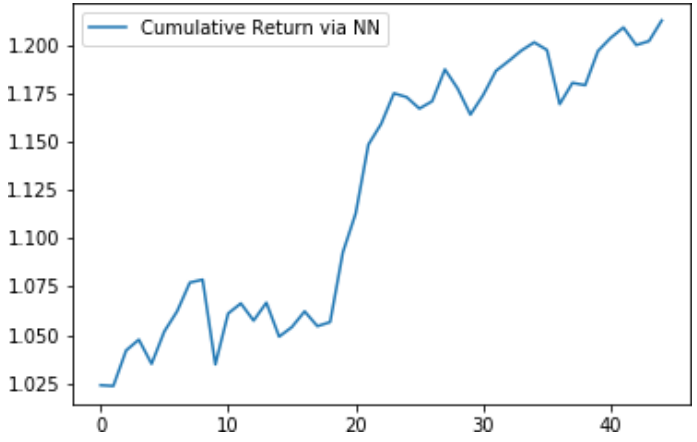


Figure: 神经网络构建投资组合的表现

目录

1 前馈神经网络和反向传播算法

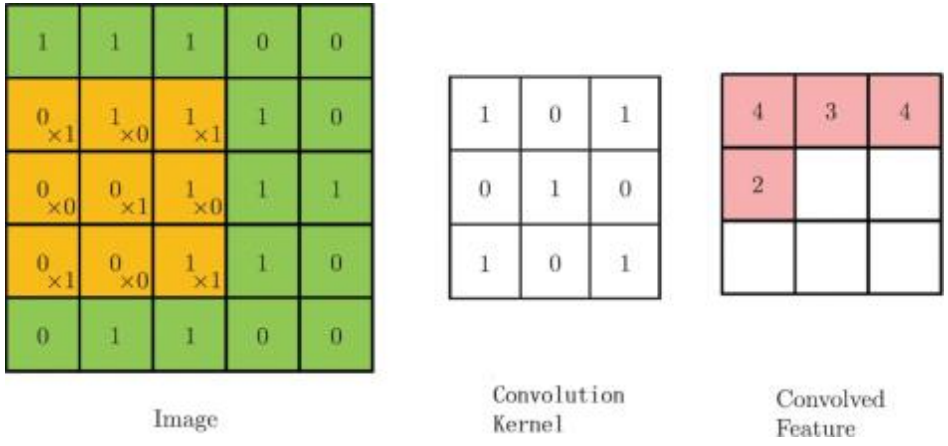
2 网络结构

3 自编码和生成模型

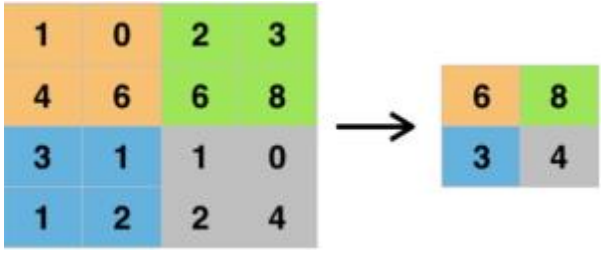
4 揭开深度学习的黑箱?

卷积神经网络 CNN

- 广泛应用于图像，视频，音频和文本数据，
- CNN 区别于其他神经网络模型的主要特征是卷积运算操作。
- 卷积运算



- 池化



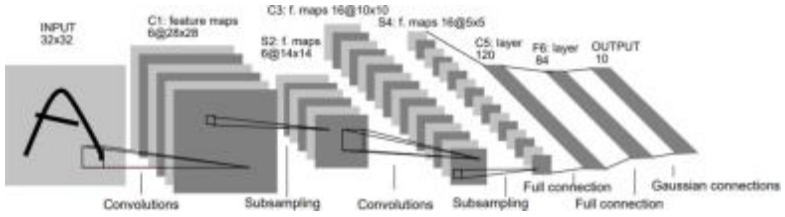
卷积神经网络 CNN

- 卷积结构

  卷积神经网络一般包含输入层、卷积层、激活函数层、池化层、标准化层、全连接层、输出层基础层级结构

  - 卷积层 -激活函数层
  - 卷积层 -激活函数层 -池化层
  - 卷积层 -激活函数层 -池化层 -标准化层

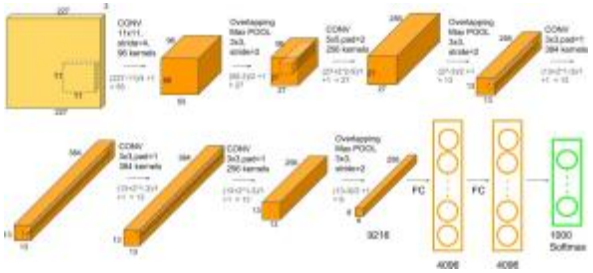
- LeNet



输入 -卷积层 -池化层 -卷积层 -池化层 -卷积层 -全连接 -输出

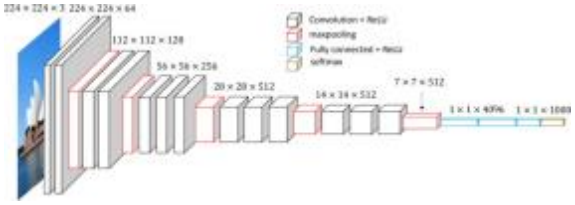
卷积神经网络 CNN

- AlexNet(2012)



AlexNet 具有 8 个模块，有 5 个卷积模块和 3 个全连接模块。

- VGGNet(2014)



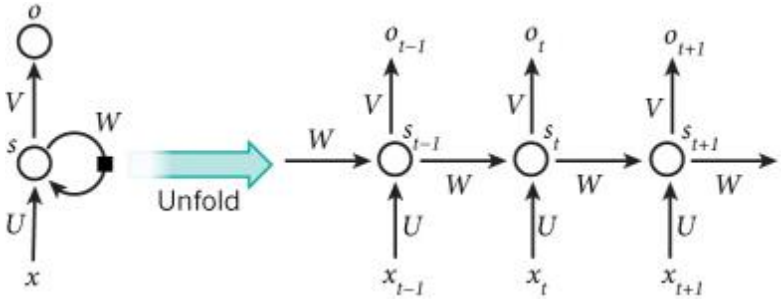
16 层 VGGNet 的网络结构，里面所有的卷积核都使用 3*3 的大小，池化层都使用大小为 2*2，步长为 2 的最大池化。

循环网络 RNN

- 用于处理序列输入和序列输出的特殊神经网络。
- 在文本生成、图像描述生成、机器翻译、语音识别方面的应用中取得了很大的成功。
- Elman network:

$$
\begin{aligned}
s_t &= \sigma_s(Us_{t-1} + Wx_t) \\
o_t &= \sigma_o(Vs_t)
\end{aligned}
$$

t 时刻的隐状态依赖于$t-1$ 时刻的隐状态 s_{t-1}。



循环网络 RNN

- Jordan network

$$s_t = \sigma_s(Uo_{t-1} + Wx_t)$$
$$o_t = \sigma_o(Vs_t)$$

隐含层 t 时刻的隐状态依赖于$t-1$ 时刻的输出 o_{t-1}。
（区别于 Elman network，依赖于 $t-1$ 时刻的隐状态 s_{t-1}）

- Minimal gated unit(Gated Recurrent Unit (GRU) 的简化形式)：

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f o_{t-1} + b_f) \\ o_t &= f_t \circ o_{t-1} + (1 - f_t) \circ \sigma_o(W_h x_t + U_t(f_t \circ o_{t-1}) + b_h) \end{aligned}$$

f_t 是 t 时刻的遗忘门，依赖于 $t-1$ 时刻的输出 o_{t-1}

- Fully gated unit(GRU 的标准形式)：

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f o_{t-1} + b_f) \quad r_t \\ &= \sigma_g(W_r x_t + U_r o_{t-1} + b_r) \quad o_t \\ &= f_t \circ o_{t-1} + (1 - f_t) \circ \sigma_o(W_h x_t + U_t(r_t \circ o_{t-1}) + b_h) \end{aligned}$$

遗忘门 f_t 又称为更新门。多了一个重设门reset gate r_t

循环网络 RNN

- LSTM(长短记忆单元):

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f o_{t-1} + b_f) \\
r_t &= \sigma_g(W_r x_t + U_r o_{t-1} + b_r) \\
c_t &= f_t \circ c_{t-1} + r_t \circ \sigma_c(W_c x_t + U_c o_{t-1} + b_h) \\
o_t &= \sigma_g(W_o x_t + U_o o_{t-1} + b_o) \circ \sigma_h(c_t)
\end{aligned}
$$

其中，f_t 是遗忘门，r_t 是输入控制，c_t 是记忆细胞
$\sigma_g(W_o x_t + U_o o_{t-1} + b_o)$ 代表输出控制，σ_g、σ_c 和 σ_h 是激活函数

- Peephole LSTM

将遗忘门、输入控制、输出控制和记忆细胞中的 o_{t-1} 替换为 c_{t-1}

Dropout 和批标准化

- Dropout
  - 解决过拟合问题。
  - 在梯度下降训练的过程中，随机舍弃部分神经元，更新参数
  - 类似于训练多个不同的神经网络并取平均预测值。
    类比于集成模型，如 bagging，random forest
- 批标准化（Batch Normalization，BN）
  - 用于提高神经网络的稳定性和效果。
  - 对每个隐藏层的输入进行标准化

$$\mathbf{x}^* = (\mathbf{x} - \mu_x)/\sigma_x$$

$$BN(\mathbf{x}) = \alpha\mathbf{x}^* + \beta$$

    通常建议把批标准化放置在激活函数 $\sigma(\cdot)$ 之前的位置，
    即将原来的 $z = \sigma(\mathbf{w}^T\mathbf{x} + b)$ 变为 $z = \sigma(BN(\mathbf{w}^T\mathbf{x} + b))$。
  - 具有类似正则化和平滑的功能。
- BN 效果类似或者优于 dropout，一般使用 BN 不要需再使用 dropout

残差网络

- 退化:
  网络层数增加时，参数更多模型更复杂，但训练误差会变大。
  这种现象称为退化，不是过拟合导致的。
- 深度残差网络主要思想:
  拟合目标映射 $H(\mathbf{x})$ 与输入的残差 $F(\mathbf{x}) = H(\mathbf{x}) - \mathbf{x}$，而非直接求目标映射。
- 一般形式:

$$H(\mathbf{x}) = F(W, \mathbf{x}) + \mathbf{x}$$

  输入和输出的维度不一致，可以引入线性投影矩阵 U:

$$H(\mathbf{x}) = F(W, \mathbf{x}) + U\mathbf{x}$$

- 是较浅网络集成学习
  一定程度上规避了梯度消失带来的问题。
- AlphaGo Zero，使用具有 19 个或 39 个残差单元的卷积残差网络。

目录

1 前馈神经网络和反向传播算法

2 网络结构

3 自编码和生成模型

4 揭开深度学习的黑箱?

自编码和生成模型

- 无监督学习：X 的分布 $f_X(x)$ 相关的量或特征
  - 低维：
    - 估计 $f_X(x)$，从中抽样生成新的数据，分布接近原始分布。
    - 如核密度估计。
  - 高维：
    - 引入一些结构表示数据的特征，在特征基础上生成新数据。
    - 如 PCA 。
    - 层级结构的神经网络在特征学习中表现出更强的能力，因此基于神经网络的生成模型具有更好的表现。
- 深度无监督学习：具有深度网络结构的无监督学习
  - 包括受限波尔兹曼机和深度信念网络，各种自编码器，表示学习/特征学习，生成模型等。

自编码

- 具有重构功能的神经网络
  将输入映射到隐含层，并用隐含层重构输入数据。
  隐含层又称为 code 层，能表示输入数据的某些特征
- 编码：从输入到隐含层的映射

$$h_j = \sigma(\mathbf{w}_j \mathbf{x}^T), j = 1, 2, \cdots, q$$

- 解码：隐含层到输出重构的映射

$$Z_j = \sigma(\mathbf{b}_j \mathbf{h}^T), j = 1, 2, \cdots, p$$

- 极小化重构误差

$$\ell(\mathbf{w}, \mathbf{b}) = \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{z}_i)$$

记 $f : \mathbf{x} \rightarrowtail \mathbf{h}$，$g : \mathbf{h} \rightarrowtail \mathbf{z}$，则自编码目标函数：

$$L(\mathbf{x}, g(f(\mathbf{x})))$$

自编码

- 有多个隐含层的自编码器
  优势类似于多层神经网络对单层神经网络的优势。直接通过梯度下降法或堆栈的方式逐层学习。
- 降噪自编码
  在编码过程中对输入加入噪音，产生含有噪音的输入 $\mathbf{x}$。

  目的：避免模型过拟合，改善泛化能力。
  目标函数：

$$L(\mathbf{x}, g(f(\mathbf{x})))$$

- 稀疏自编码
  对自编码的隐含层 $\mathbf{h}$ 加入惩罚项，使得隐含层中尽量少的单元被激活。
  目标函数：

$$L(\mathbf{x}, g(f(\mathbf{x}))) + P(f(\mathbf{x}))$$

  其中 $P(f(\mathbf{x}))$ 是对隐含层 $\mathbf{h} = f(\mathbf{x})$ 的惩罚。

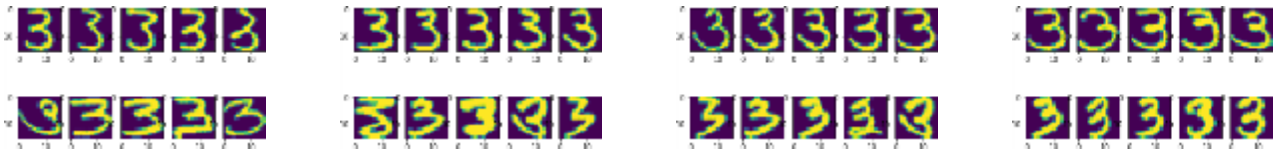
自编码 -例

- 目标和背景

  对手写数据集"zip.train" 中手写数字 3 进行特征分析
  使用自编码器对比主成分分析

- 解决方案和程序

  $f(\mathbf{x})$ 对应主成分分析的 $\mathbf{V}^T(\mathbf{x}-\bar{\mathbf{x}})$,
  $g(f(\mathbf{x}))$ 对应主成分分析的 $\bar{\mathbf{x}}+\mathbf{V}\mathbf{V}^T(\mathbf{x}-\bar{\mathbf{x}})$。
  自编码中隐含层某个单元数值与主成分分析中的投影得分对应
  不同：主成分使用了线性变换，而自编码使用了非线性变换。



(a)　(b)　(c)　(d)

第一个隐含层单元：宽 -窄特征，对应第一主成分。
第三个隐含层单元：长翘尾特征，对应第四主成分。
自编码的神经网络训练过程有一定随机性，在特征提取上不稳定

逐层特征学习

- 多层神经网络引入 ReLU，批标准化等技术之前很难训练
  逐层特征学习是深度神经网络参数初始化和预训练方法
- 受限波尔兹曼机 RBM+ 贪婪逐层训练：
  得到堆栈式的 RBM 网络，即深度信念网络
- 自编码器 + 逐层特征学习：
  类似栈式 RBM 网络，自编码器逐层训练，得到栈式自编码
  (1) 训练自编码器，得到隐含层，
  (2) 将隐含层当作自编码器输入，得到第二个隐含层。
  - 如此重复逐层学习原始数据的特征。

  变体：如堆栈降噪自编码等。

逐层特征学习

- 引入 ReLU，批标准化后

  不需要通过逐层学习的方式来获得初始化参数，
  但在一些缺少标签的数据中，逐层特征学习仍是种可行的方法。
- 通过反卷积方法，通过可视化发现深度卷积神经网络中各个隐含层代表原始输入不同层次的特征，后一层特征比前一层更抽象。
- 在监督学习中，很多时候高级的特征对类别预测有很大帮助，深度学习在特征提取上相对于浅层学习具有天然优势。

生成对抗网络 GAN

- 学习数据的分布的新方法，可以生成非常逼真，接近真实的图像
- 思想：生成器和判别器两个神经网络彼此博弈。
- 生成网络的优化目标（给定判别网络 D_θ）：

$$\min_{\phi} \mathrm{E}_{\mathbf{z}\sim p_z(z)} \log(1 - D_\theta(G_\phi(\mathbf{z})))$$

判别网络的优化目标 (给定生成网络 G_ϕ):

$$\max_{\theta} \mathrm{E}_{\mathbf{x}\sim p_x(\mathbf{x})} \log(D_\theta(\mathbf{x})) + \mathrm{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_\theta(G_\phi(\mathbf{z})))$$

统一的形式的优化目标:

$$\min_{\phi} \max_{\theta} \mathrm{E}_{\mathbf{x}\sim p(x)} \log(D_\theta(\mathbf{x})) + \mathrm{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_\theta(G_\phi(\mathbf{z})))$$

- 推广模型：如 DCGAN，条件 GAN，f-GAN，pBig-GAN，Wasserstein-GAN，Bidirectional GAN, 等。

变分自编码 VAE

- 基于自编码的生成模型，在图像生成和半监督学习中有很好表现
- 编码表达

$$p_\theta(\mathbf{x}, \mathbf{h}) = p_\theta(\mathbf{h}|\mathbf{x})p(\mathbf{x})$$

$p_\theta(\mathbf{x}|\mathbf{h})$ 是一个具有神经网络结构的条件分布，如多元正态分布

- 解码表达

$$q_\phi(\mathbf{x}, \mathbf{h}) = q_\phi(\mathbf{x}|\mathbf{h})q(\mathbf{h})$$

类似的，$q_\phi(\mathbf{x}|\mathbf{h})$ 是一个具有神经网络结构的条件分布

- 求解：最小化 $q_\phi(\mathbf{x}, \mathbf{h})$ 到 $p_\theta(\mathbf{x}, \mathbf{h})$ 的 KL 距离

变分自编码 VAE

- 最小化

$$
\begin{aligned}
& KL(p_\theta(\mathbf{h}|\mathbf{x})p(\mathbf{x})||q_\phi(\mathbf{x}|\mathbf{h})q(\mathbf{h})) \\
= \quad & \mathbf{E}_{\mathbf{x}\sim p(\mathbf{x})} \int p_\theta(\mathbf{h}|\mathbf{x}) \log \frac{p(\mathbf{x})p_\theta(\mathbf{h}|\mathbf{x})}{q_\phi(\mathbf{x}|\mathbf{h})q(\mathbf{h})} d\mathbf{h} \\
= \quad & \mathbf{E}_{\mathbf{x}\sim p(\mathbf{x})} \left[-\mathbf{E}_{\mathbf{h}\sim p_\theta(\mathbf{h}|\mathbf{x})} \log q_\phi(\mathbf{x}|\mathbf{h}) + KL(p_\theta(\mathbf{h}|\mathbf{x})||q(\mathbf{h})) \right] + constant
\end{aligned}
$$

第一项：从条件分布 $p_\theta(\mathbf{h}|\mathbf{x}_i)$ 抽样，代入计算 $\log q_\phi(\mathbf{x}_i|\mathbf{h}^*)$ 导数。
第二项：假设 $q(\mathbf{h})$ 是标准多元正态分布，对第二项化简求导数。

- 最小化第一项：最小化重构误差，
  最小化第二项：编码模型生成的隐变量的分布接近标准多元正态
- 这两项是有对抗意味的

Recommend 火龙果软件

目录

1 前馈神经网络和反向传播算法

2 网络结构

3 自编码和生成模型

4 揭开深度学习的黑箱?

Recommend
火龙果软件

揭开深度学习的黑箱?

- 深度神经网络和深度学习的成功涵盖了监督学习，无监督学习和强化学习三个主流的机器学习领域。
  - 监督学习: 深度卷积网络，循环神经网络，
  - 无监督学习: 生成对抗网络，变分自编码，
  - 强化学习:Alpha Go 等，
- 为何深度学习有效?

Recommend
火龙果软件

特征提取和降维

- 数据维数不高，数据量不大时，浅层学习足够。
  高维和大量数据，如处理图像这种高维数据时，深度学习。
- 深度学习方法突出优点：
  同时兼备了非线性建模、特征提取和降维三个特点
- 深度学习是如何实现特征提取的?

Recommend
火龙果软件

信息瓶颈

- 万能近似定理：
  具有一个隐含层的前馈神经网络可以逼近一个（紧集上）任意连续函数。
- 但是，池化，dropout，降噪自编码中给输入数据增加噪音。为什么在限制神经网络的能力?

Recommend
火龙果软件

信息瓶颈

- 信息瓶颈理论：
  提供了一种深度学习的解释，即控制层与层之间流动的信息量
- 以变分自编码为例

$$\mathbf{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[-\mathbf{E}_{\mathbf{h}\sim p_\theta(\mathbf{h}|\mathbf{x})}\log q_\phi(\mathbf{x}|\mathbf{h}) + KL(p_\theta(\mathbf{h}|\mathbf{x})||q(\mathbf{h}))\right]$$

* 第一项代表重构误差，是编码模型生成隐含层并代入到解码模型的似然
  可以写作 $I(\mathbf{x}, \hat{\mathbf{x}})$，其中 $\hat{\mathbf{x}}$是 $\mathbf{x}$ 的某种表
* 第二项可以近似看成是信息量 $\mathrm{E}[KL(p_\theta(\mathbf{h}|\mathbf{x})||p(\mathbf{h}))]$ 的一个上界，它的作用是控制流入隐含层的信息量。可以写作 $I(\mathbf{x}, \mathbf{h})$。

Recommend
火龙果软件

信息瓶颈

- Beta-变分自编码（Beta-VAE）

$$-\mathbf{E}_{\mathbf{h}\sim p_\theta(\mathbf{h}|\mathbf{x}_i)} \log q_\phi(\mathbf{x}_i|\mathbf{h}) + \beta KL(p_\theta(\mathbf{h}|\mathbf{x}_i)||q(\mathbf{h}))$$

用到监督学习，如变分信息瓶颈：

$$\frac{1}{N}\sum_{i=1}^{N}\left[-\mathbf{E}_{\mathbf{h}\sim p_\theta(\mathbf{h}|\mathbf{x}_i)} \log q_\phi(\mathbf{y}_i|\mathbf{h}) + \beta KL(p_\theta(\mathbf{h}|\mathbf{x}_i)||q(\mathbf{h}))\right]$$

变分自编码的目标函数与变分信息瓶颈的目标函数是一致的

$$I(\mathbf{x};\hat{\mathbf{x}}) - \beta I(\mathbf{y};\hat{\mathbf{x}})$$

* 第一项是极小化基于特征表示（隐含层）的重构误差，
第二项用来控制流入隐含层的信息量。
* 一个好的表示应能保留 **x** 中与 **y** 相关信息，抛弃 **x** 中与预测 **y** 不相关信息。 在信息瓶颈的模式中，神经网络的每一层都可以控制信息量的流入，因此信息量（隐含层表示与标签的互信息）从输入到输出将逐渐递减。
* 信息瓶颈的研究还给出了信息量估计的上界，以及神经网络性能度量。

Recommend
火龙果软件

结语

- 深度学习在多个领域取得巨大成功
- 深度学习通过多层神经网络逐层抽象，实现自动特征提取
- 深度神经网络同时具有特征提取、非线性建模和降维的能力
- 信息瓶颈：一些网络设计和训练技巧限制了层间信息流动量
- 理论方面期待更多的研究成果

Recommend
火龙果软件