

卷積神經網路 (八) 訓練CNN



山與水你和我

體系結構小白，影像處理小白

13 人贊同了該文章

上一篇

[山與水你和我：卷積神經網路 \(七\) 搭建CNN 網路結構](#)

定義好了網路結構，資料在

[山與水你和我：卷積神經網路 \(二\) 從影像到tensor](#)

也準備好了，還差最後一步，設定損失函數，朝著最小化損失函數的方向做梯度下降。在影像分類任務中，通常使用**softmax + 交叉熵**的組合。

軟最大

上一篇定義網路結構中，最後的輸出層是一個Linear 層，將Linear 層的輸出作為softmax 的輸入。

為了簡單，先拿batch_size = 1 做例子。

向前

假設softmax 層的輸入向量為 **K** 維的 **z** 向量，即

$$q_i = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{K-1} e^{z_j}}, \quad i = 0, 1, \dots, K-1$$

輸出的 **K** 個值之和為1，可以代表特定的機率分佈，分別代表 **K** 類的機率。

落後

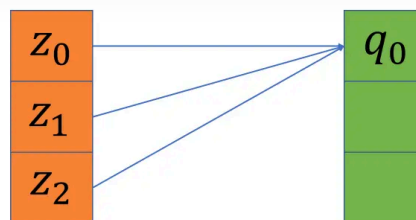
要backward，求對softmax 輸入的梯度，就要找softmax 每個輸入都參與了什麼計算，對哪些輸出做出了貢獻，如下圖例子

登入即可查看**超5億**專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。

立即登入/註冊





$$q_0 = \frac{e^{z_0}}{e^{z_0} + e^{z_1} + e^{z_2}}$$

👍👍👍👍👍

這裡計算softmax 層輸出 q 對輸入 z 的導數 $\frac{\partial q_i}{\partial z_j}$ (因為每一個 q_i 都受每一個 z_j 的影響)

$$\frac{\partial q_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{i=0}^{K-1} e^{z_i}}}{\partial z_j} = \frac{(e^{z_i})' \sum_{i=0}^{K-1} e^{z_i} - (\sum_{i=0}^{K-1} e^{z_i})' e^{z_i}}{(\sum_{i=0}^{K-1} e^{z_i})^2}$$

為了更好地化簡，分情況討論

• 我=j

此時， $(e^{z_i})' = e^{z_i}$ ， $(\sum_{i=0}^{K-1} e^{z_i})'$ 也只有第j 項導數 e^{z_j} 可能不為0，因此

$$\begin{aligned} \frac{\partial q_i}{\partial z_j} &= \frac{e^{z_i} \times \sum_{i=0}^{K-1} e^{z_i} - e^{z_i} \times e^{z_j}}{(\sum_{i=0}^{K-1} e^{z_i})^2} \\ &= \frac{e^{z_i}}{\sum_{i=0}^{K-1} e^{z_i}} \times \frac{\sum_{i=0}^{K-1} e^{z_i} - e^{z_j}}{\sum_{i=0}^{K-1} e^{z_i}} \\ &= \frac{e^{z_i}}{\sum_{i=0}^{K-1} e^{z_i}} \times \left(1 - \frac{e^{z_j}}{\sum_{i=0}^{K-1} e^{z_i}}\right) \\ &= q_i \times (1 - q_j) \\ &= q_i \times (1 - q_i) \end{aligned}$$

• 我≠j

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



$$\begin{aligned}
 \frac{\partial q_i}{\partial z_j} &= \frac{0 - e^{z_j} \times e^{z_i}}{(\sum_{i=0}^{K-1} e^{z_i})^2} \\
 &= - \frac{e^{z_j}}{\sum_{i=0}^{K-1} e^{z_i}} \times \frac{e^{z_i}}{\sum_{i=0}^{K-1} e^{z_i}} \\
 &= -q_j \times q_i
 \end{aligned}$$

因此，可得softmax 任一輸出 y_i 對任一輸入 z_j 的導數為

$$\frac{\partial q_i}{\partial z_j} = \begin{cases} q_i \times (1 - q_i), & i = j \\ -q_j \times q_i, & i \neq j \end{cases}$$

交叉熵

網路輸出經過softmax 得到歸一化的機率，目標就是使得這個機率分佈與目標機率分佈越接近！常用的有KL 散度。

KL 散度

KL 散度，也稱為**相對熵**，在機器學習中，常用來衡量：對於同一個隨機變量 x ，**樣本真實分佈 p** 和**預測分佈 q** 之間的差異。假設隨機變數 x 有 K 維，衡量以上兩個分佈的KL 散度，公式如下：

$$D_{KL}(p, q) = \sum_{i=0}^{K-1} p_i \log_2 \left(\frac{p_i}{q_i} \right)$$

對上式做拆分，得到

$$\begin{aligned}
 D_{KL}(p, q) &= \sum_{i=0}^{K-1} p_i \log_2 \left(\frac{p_i}{q_i} \right) \\
 &= \sum_{i=0}^{K-1} p_i \log_2(p_i) - \sum_{i=0}^{K-1} p_i \log_2(q_i) \\
 &= -H(p) - \sum_{i=0}^{K-1} p_i \log_2(q_i) \\
 &= -H(p) + H(p, q)
 \end{aligned}$$

其中，前一項是真實分佈 p 的熵，是個常數；後一項 $H(p, q)$ 就是交叉熵，一般影像分類中優化的是交叉熵，等價於最小化分佈 p, q 之間的KL 散度。

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



給定單一樣本的真實分佈 p 和預測分佈 q ，交叉熵定義為

$$H(p, q) = - \sum_{i=0}^{K-1} p_i \log_2(q_i).$$

落後

目標函數是最小化交叉熵，其中預測分佈的 $q_i = \frac{e^{z_i}}{\sum_{j=0}^{K-1} e^{z_j}}$ 是透過softmax得到的，前面算過了

softmax 任一輸出 y_i 對任一輸入 z_j 的導數，在這個導數基礎上，直接算交叉熵對 z_j 的導數：

$$\begin{aligned} \frac{\partial H(p, q)}{\partial z_j} &= - \sum_{i=0}^{K-1} \frac{\partial p_i \log(q_i)}{\partial z_j} \\ &= - \sum_{i=0}^{K-1} \left(\frac{\partial p_i \log(q_i)}{\partial q_i} \times \frac{\partial q_i}{\partial z_j} \right) \\ &= - \sum_{i=0}^{K-1} \left((p_i \times \frac{\partial \log(q_i)}{\partial q_i}) \times \frac{\partial q_i}{\partial z_j} \right) \\ &= - \sum_{i=0}^{K-1} \left(p_i \times \frac{1}{q_i} \times \frac{\partial q_i}{\partial z_j} \right) \text{开始代入} \\ &= -p_j \times \frac{1}{q_j} \times q_j(1 - q_j) - \sum_{i=0, i \neq j}^{K-1} p_i \times \frac{1}{q_i} \times (-q_i \times q_j) \\ &= -p_j \times (1 - q_j) + \sum_{i=0, i \neq j}^{K-1} p_i \times q_j \\ &= -p_j + q_j \times p_j + \sum_{i=0, i \neq j}^{K-1} q_j \times p_i \\ &= -p_j + q_j \times (p_j + \sum_{i=0, i \neq j}^{K-1} p_i) \\ &= -p_j + q_j \times 1 \\ &= q_j - p_j \end{aligned}$$

(註：為了方便求導，這裡的 \log 指的是 \ln ，和交叉熵裡的 \log_2 不同。但二者的導數就只差一個常數—— $(\log_2 x)' = \frac{(\ln x)'}{\ln 2} = \frac{1}{x} \cdot \frac{1}{\ln 2}$ ，所以不影響優化過程)

登入即可查看超5億專業優質內容

超5千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



上面寫的是一個樣本的情況。對於所有的樣本，最小化機率分佈和真實分佈的**交叉熵總和**，就可以達到學習特定資料集分佈的效果。

實現

前面兩個結合挺簡單，但實作還是有一些細節。

- exp 溢出的問題

由於CNN 最後一層輸出是Linear 層，沒有用sigmoid、tanh 等函數限定輸出值的取值範圍，因此很容易溢出，為了盡量緩解這種情況，一般有兩種做法① 輸出層加sigmoid 或者tanh，但這兩個激活函數都有容易飽和的問題，不建議② softmax 上下除以一個常數，值大小不變，一般設softmax 的輸入最大值為 M ，softmax 公式上下除以 e^M ，如下

$$\begin{aligned} q_i = \text{softmax}(z_i) &= \frac{e^{z_i} / e^M}{\sum_{j=0}^{K-1} (e^{z_j} / e^M)} \\ &= \frac{e^{z_i - M}}{\sum_{j=0}^{K-1} e^{z_j - M}} \end{aligned}$$

softmax 結果是不變的，舉個例子：網路輸出 $z = [5000, 5030, 5040]$ ，直接算e 的對數，會溢出，但用上面的步驟，就變成了對 $[-40, -30, 0]$ 算softmax，因為最大值就是0，更不容易溢出，，結果還一樣。

但還是存在一點瑕疵，最大值為0，但最小值沒有限制， e^{-5000} 這種其實可以省去計算，直接看成0。

個人是這樣寫的

```
inline data_type __exp(const data_type x) {
    if(x >= 88) return FLT_MAX; // 直接返回 float 的最大值，如果 data_type 換成 double 則
    else if(x <= -50) return 0.f;
    return std::exp(x);
}
```

softmax 實作如下

登入即可查看**超5億**專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



```
const int num_classes = input[0]->get_length();
std::vector<tensor> output;
output.reserve(batch_size);
for(int b = 0; b < batch_size; ++b) {
    tensor probs(new Tensor3D(num_classes));
    // 首先算出输出的最大值, 防止溢出
    const data_type max_value = input[b]->max();
    data_type sum_value = 0;
    for(int i = 0; i < num_classes; ++i) {
        probs->data[i] = __exp(input[b]->data[i] - max_value); // 减去最大值
        sum_value += probs->data[i];
    }
    // 概率之和 = 1
    for(int i = 0; i < num_classes; ++i) probs->data[i] /= sum_value;
    // 去掉一些 nan
    for(int i = 0; i < num_classes; ++i) if(std::isnan(probs->data[i])) probs->data[i] = 0;
    output.emplace_back(std::move(probs));
}
return output;
}
```

- one-hot 產生真實分佈 p

單一樣本，計算交叉熵需要預測分佈 q ，也就是softmax 的輸出；還需要樣本的真實分佈 p ，在影像分類中，知道的是類別的序號，從0 開始，到 $K - 1$ ，舉個例子，假設有五分類，某圖片樣本的類別序號是2，則它的真實分佈 p 可以表達成 $[0, 0, 1, 0, 0]$ ，這個過程稱作one-hot，獨熱編碼。

one-hot 是本實驗採用的編碼方式，也是硬編碼，樣本屬於某一類的機率為1，屬於其他類別的機率為0。

也存在一些其他編碼，如soft label，例如有0.9 的機率是屬於標籤所在類，剩餘0.1 的機率均攤給其他類，這樣的做法可以起到緩解過擬合的效果，避免過度相信訓練樣本，如果有噪音的話。

還有在知識蒸餾中，student 網路使用的真實分佈標籤可以是teacher 網路的預測結果，一張圖像可以屬於比熊犬的機率是0.4，屬於愛斯基摩犬的機率是0.25，屬於哈士奇的機率是0.1等，似是而非，有更合理。

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



優化的目標函數準備好了，即可模仿Pytorch 寫CNN 的訓練過程。

最佳化器

優化目標函數一般可以用梯度下降法，更新參數。但有幾個問題：①每次更新多少②只看目前樣本的梯度？要不要考慮歷史樣本的梯度？如下

① 設定學習率，在CNN章節的update_gradients 中提過，參數 w 的更新如下：

$$w = w - lr * \delta$$

lr 是學習率，一般設的比較小，0.001 甚至0.0001。學習率太高，每次更新的太多，目標函數容易振盪；學習率太小訓練時間漫長，容易陷入局部最優解，難辦，我就不辦了，不懂。

② 只考慮目前訓練的樣本回饋的訊息，也就是SGD 方法，不夠穩定，例如遇到一些異常樣本，使得優化方向偏離理想的方向，很玄學，常用的改進版本有mini-batch SGD、momentum 動量、Adagrad、RMSProp、Adam等優化器，都會藉助歷史樣本的最佳化資訊幫助矯正。為了簡單，本實驗暫時只實現了mini-batch SGD，暫時沒有想到比較優雅的解決方案。

實現

主要訓練程式碼模仿的Pytorch 寫法，如下

```
using namespace architectures;

// 指定一些参数
const int train_batch_size = 4;
const int valid_batch_size = 1;
const int test_batch_size = 1;
assert(train_batch_size >= valid_batch_size and train_batch_size >= test_batch_size);
assert(valid_batch_size == 1 and test_batch_size == 1); // 设计问题，暂时只支持这个
const std::tuple<int, int, int> image_size({224, 224, 3});
const std::filesystem::path dataset_path("../datasets/animals");
const std::vector<std::string> categories({"dog", "panda", "bird"});

// 获取图片
auto dataset = pipeline::get_images_for_classification(dataset_path, categories);

// 构造数据流
pipeline::DataLoader train_loader(dataset["train"], train_batch_size, true, true, imag
```

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



```
// 定义网络结构
const int num_classes = categories.size(); // 分类的数目
AlexNet network(num_classes, false);

// 直接加载
// network.load_weights("../checkpoints/AlexNet_aug_2e-4/iter_100000_train_0.846_valia

// 保存
const std::filesystem::path checkpoints_dir("../checkpoints/AlexNet_aug_1e-3");
if(not std::filesystem::exists(checkpoints_dir))
    std::filesystem::create_directories(checkpoints_dir);
std::filesystem::path best_checkpoint; // 当前正确率最高的模型
float current_best_accuracy = -1; // 记录当前最高的正确率

// 开始训练
const int start_iters = 1; // 从第几个 iter 开始
const int total_iters = 400000; // 训练 batch 的总数
const float learning_rate = 1e-3; // 学习率
const int valid_inters = 1000; // 验证一次的间隔
const int save_iters = 5000; // 保存模型的间隔
float mean_loss = 0.f; // 平均损失
float cur_iter = 0; // 计算平均损失用的
ClassificationEvaluator train_evaluator; // 计算累计的准确率
std::vector<int> predict(train_batch_size, -1); // 存储每个 batch 的预测结果, 和 labels
// 开始训练
for(int iter = start_iters; iter <= total_iters; ++iter) {
    // 从训练集中采样一个 batch
    const auto sample = train_loader.generate_batch();
    // 送到网络中
    const auto output = network.forward(sample.first);
    // 网络输出经过 softmax 转化成概率
    const auto probs = softmax(output);
    // 输出概率和标签计算交叉熵损失, 返回损失项和梯度
    auto loss_delta = cross_entropy_backward(probs, one_hot(sample.second, num_classes));
    mean_loss += loss_delta.first;
    // 根据损失, 回传梯度
    network.backward(loss_delta.second);
    // 更新权值
    network.update_gradients(learning_rate);
    // 根据 predict 和 label 计算准确率
    for(int b = 0; b < train_batch_size; ++b) predict[b] = probs[b]->argmax(); // 概率最
    train_evaluator.compute(predict, sample.second);
    // 打印信息
```

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。




```
(cuda_10.1_python_3.7) PS D:\work\crane\deep_learning\cnn\cpu\src> ./run.exe
```

如上圖，損失一直在下降，正確率基本上保持著上升的趨勢，說明應該是寫對了！處理器i5-10400f，單核心跑，影像尺寸224 * 224，速度感覺還行。cuda 版本還在出發的路上。


驗證和測試階段，跟train 不一樣的主要有兩點：

- Batch norm 和Dropout 等手段在非訓練階段有所變化，對應Pytorch 中的`.train()`、`.eval()`函數
- 非訓練階段，可以不記錄一些輔助反向傳播的變量，在推理（測試）階段更是可以避免開闢這些變量的空間，減少空間佔用，同時加快運算，對應Pytorch 中的`no_grad()`函數。整個一個流程下面，確實`no_grad()` 可以大幅減少空間的佔用，訓練階段，很大一部分空間都是用於backward 的。

程式碼

所有程式碼，包括資料集都放在

<https://github.com/hermosayhl/CNN>

號  github.com/hermosayhl/CNN

編輯於2022-02-28 15:06

深度學習 (Deep Learning)

卷積神經網路 (CNN)

C++

號  贊同13

號  ▼

號  7 則評論

號  分享

號  喜歡

號  收藏

號  申請轉載

號  ...

寫下你的評論...

7 則評論

預設

最新

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



八倍，那就不這個函數裡面去寫應該有兩個循環才對啊，那不就O(N^2)是了，那就不可能圖片肯定比4大，不應該還有一個循環來遍歷所有的圖片嗎😂

2022-04-12

號 ● 回覆 號 ♥ 喜歡

**山與水你和我** 作者

我程式還有部分沒貼出來，就每隔幾個batch 之後，重新計算損失函數和準確率這些

2022-04-12

號 ● 回覆 號 ♥ 喜歡

**FuckU 類別 (物件)** ▸ **山與水你和我**

哦哦好的，不過大佬方便留個聯絡方式嗎，我也在搞一個c++的機器學習庫，CUDA的輪子我寫好了😂最近在構建網絡

2022-04-12

號 ● 回覆 號 ♥ 喜歡

展開其他2 則回復號 >

**愛cv**

大佬，方便留聯絡方式，運行這個cnn工程，報了的一堆錯誤😂

2022-03-29

號 ● 回覆 號 ♥ 喜歡

**山與水你和我** 作者

好

2022-03-29

號 ● 回覆 號 ♥ 喜歡

文章被以下專欄收錄



電腦視覺基礎

最簡單的卷積神經網路、ReLU、BN、目標偵測等

推薦閱讀

卷積神經網路的訓練過程

卷積神經網路的訓練過程卷積神經網路的訓練過程分為兩個階段。第一個階段是資料由低層次傳播到高層次的階段，即前向傳播階段。另外一個階段是，當前向傳播得出的結果與預期不相符時，將誤差...

元沫

卷積神經網路的改進

提升卷積神經網路的技巧主要包括幾個面向：1) 資料增強；2) 影像預處理；3) 網路的初始化；4) 訓練期間的小技巧；5) 活化函數的選擇；6) 正則化策略；7) 從圖中判斷模型表現；-----...

從那裡



深度學習(8): 卷積神經網路3—訓練方法及網路結構設計

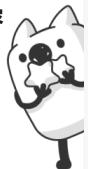
技術工匠6

卷積神經網路操作

作者| Justin h
AlexNet發展至各種各樣的CNN，一個比一個深，一個比一個輕量。我下面將變革性的工作進行簡單盤點，從...
CDA資料分析師

登入即可查看超5億專業優質內容

超5千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



知乎

首發於
電腦視覺基礎

切换模式

×

登入即可查看**超5億**專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。

