Нейронные сети для изображений

Старков Артём, гр. 622

Санкт-Петербургский государственный университет
Математико-механический факультет
Статистическое моделирование

Санкт-Петербург
2017

Постановка задачи классификации в нейросетях

Классификация на M классов. Пусть $\mathcal{X}$ – множество значений индивидов x, $\mathcal{Y}$ – множество классов y. Имеем неизвестную зависимость

$$y = f(x), f : \mathcal{X} \to \mathcal{Y}.$$

Зафиксируем $X \subset \mathcal{X}$, для которых $f(x)$ известно. Получаем задачу классификации с учителем; ищем $f^*(x)$, чтобы функционал качества

$$Q(f^*, X) = \frac{1}{m} \sum_{x \in X} \mathcal{L}(f^*(x), f(x))$$

достигал минимума на X.

Постановка задачи классификации в нейросетях

Метод минимизации эмпирического риска:

$$f^* = \arg\min_{f' \in \mathcal{F}} Q(f', X),$$

где $\mathcal{F}$ – класс функций, аппроксимирующих целевую функцию $f(x)$.
Для нейронных сетей задача решается для параметризованных аппроксиматоров $f^*(x;\theta)$ путем подбора параметров весов:

$$\min_{\theta} \frac{1}{m} \sum_{x \in X} \mathcal{L}(f^*(x;\theta), f(x)).$$

Многослойные сети: на вход следующих слоев подается выход из предыдущих через нелинейность $\sigma(.)$.

Особенности при применении к изображениям

При применении данной задачи к изображениям обнаруживается ряд особенностей, позволяющих обособить эту задачу от других:

- локальная скоррелированость пикселей;
- распределенность признака: один пиксель практически не несет в себе информации, признак на изображении – это совокупность пикселей;
- возможные трансформации детектируемых шаблонов: поворот, масштабирование, смещение, перекрытие другими предметами сцены и др.

Свертка

Свертка помогает выделить признаки из совокупности пикселей. Свертка функций f и w по области D:

$$(f * w)(x) = \int_D f(y)w(x-y)dy = \int_D f(x-y)w(y)dy;$$

Например, пусть $f : Z \times Z \to R^K, w : D \times D \to R^K, D = 1..d$ (свертка изображения по ядру w размером $d \times d$ и глубины каналов K):

$$(f * w)(i,j) = \frac{1}{Kd^2} \sum_{k=1}^{K} \sum_{t,n=1}^{d} f_k(i-t, j-n) w_k(t,n).$$

Feature map (activation map)

Пусть на вход подается изображение размером $M \times N$. Обозначим его как отображение $x : Z \times Z \to R^3$. Введем ядро свертки, представляющее собой набор весов $W : Z \times Z \to R^3$. Рассмотрим следующее отображение:

$$f(i,j) : Z \times Z \to R;$$

$$f(i,j) = \sigma((W * x)(i,j) + b);$$

где b — некоторое смещение, σ — нелинейная функция. Вектор из таких отображений, построенный на *разных* ядрах W_k и смещениях b_k называется **feature map**.

Сверточный слой

Сверточный слой:

- входная feature map
$$f^{(s-1)}(i,j) \in R^{N^{(s-1)} \times N^{(s-1)} \times K^{(s-1)}};$$
- $K^{(s)}$ обучаемых ядер
$$W \in R^{d \times d \times K^{(s-1)}};$$
- $K^{(s)}$ смещений
$$b^{(s)} \in R;$$
- выходная feature map, подаваемая на вход следующего слоя:
$$f^{(s)}(i,j) \in R^{N^{(s)} \times N^{(s)} \times K^{(s)}};$$

Внешние параметры слоя: глубина feature map $K^{(s)}$, размер ядра $d^{(s)}$, нелинейность $\sigma(.)$.

Rectifier linear unit (ReLU)

В сверточных сетях функцией активации чаще всего выбирают Rectifier linear unit (ReLU):

$$f(x) = \max(x, 0).$$

Softplus – гладкий ReLU:

$$f(x) = \ln(1 + e^x).$$

Плюсы: очень просто вычисляемая производная: $(max(0, x))' = \gamma(x > 0)$; нет проблемы vanishing gradient справа.
Минусы: проблема «мертвых нейронов»: нейрон может достичь состояния, когда градиент становится нулевым, и обучения больше не происходит.

Варианты ReLU

Варианты ReLU для избежания проблемы «мертвых нейронов»:

- Leaky ReLU ($\epsilon > 0$ – входной параметр):

$$f(x) = \begin{cases} x, & x \geq 0, \\ \epsilon x & x < 0. \end{cases}$$

- Shifted ReLU ($\alpha > 0$ – входной параметр):

$$f(x) = \max(x, 0) - \alpha.$$

- Exponential ReLU (ELU) ($a > 0$ – обучаемый параметр):

$$f(x) = \begin{cases} x, & x \geq 0, \\ a(e^x - 1) & x < 0. \end{cases}$$

Субдискретизация (pooling)

Слой пулинга (pooling): нелинейное уплотнение карты признаков. для снижения размерности и выявления новых признаков на больших расстояниях.

$$maxpool(x)(i,j) = \max\left(\{x(i+t, j+n)\}_{t,n=1}^{d}\right),$$

где $d \times d$ – размер ядра пулинга.

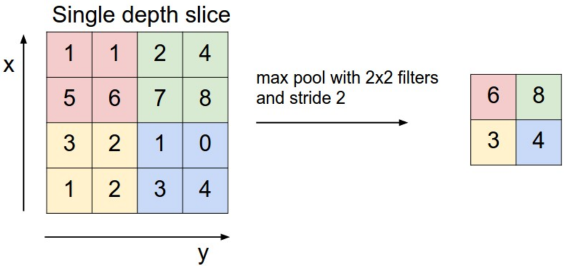

Рис.: Max pooling на одном слое с ядром 2×2

Data augmentation, DropOut

Definition

Data augmentation (увеличение данных) — техника расширения пространства обучающих индивидов (изображений) путем применения аффинных преобразований и некоторых специфических приемов, например генерация изображений. Например, поворот изображения на некоторый угол, изменение размера изображения и др.

Definition

DropOut — техника для регуляризации обычной сети: вводится новая модель нейрона, где каждый нейрон имеет вероятность p не участвовать в следующей эпохе обучения сети; p — общий внешний параметр. В рабочем режиме $p = 1$. В некотором смысле данная техника эквивалентна обучению множества нейросетей и усреднению результата.

Архитектура LeNet

Архитектура LeNet: Yann LeCun et al.(1998). Параметры:

- 2 сверточных слоя, 2 max-pooling, 3 полносвязных, 10 классов на выходе;
- 60 тыс. весов, из них 3 тыс. для сверточных слоев, остальные для полносвязных.

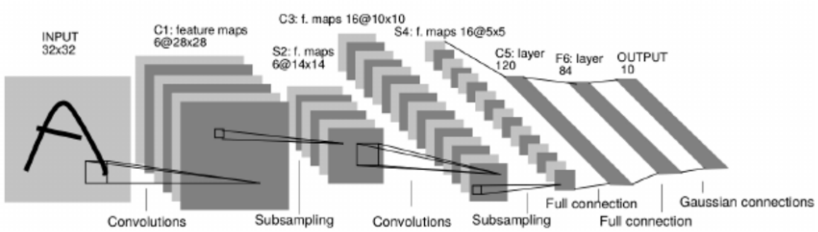

Рис.: Архитектура LeNet

Архитектура AlexNet

Архитектура AlexNet: Alex Krizhevsky et al.(2012). Параметры:

- 60 млн. весов, из них 4 млн. для сверточных слоев;
- 5 сверточных слоев, 3 max-pooling, 3 полносвязных, 1000 классов на выходе;
- ReLU, DropOut ($p = 0.5$), data augmentation.

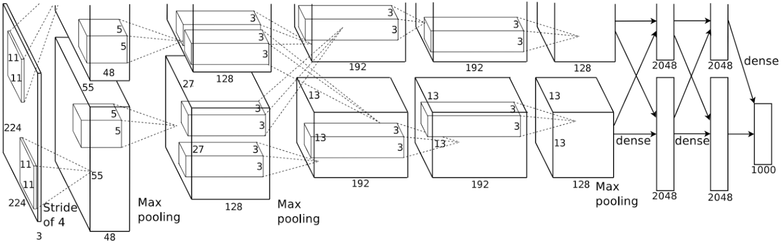

Рис.: Архитектура AlexNet

Свертка 1×1

Пусть $f(i,j) : Z \times Z \to R^K$ – сверточный слой; $x(i,j)$ – вход:

$$f(i,j) = \sigma((W * x)(i,j) + b).$$

Задача: заменить линейный классификатор более универсальным: например, n-слойным перцептроном:

$$f^{(1)}(i,j) = \sigma(W^{(1)}x(i,j) + b^{(1)}),$$
$$\dots$$
$$f^{(n)}(i,j) = \sigma(W^{(n)}f^{(n-1)}(i,j) + b^{(n)}).$$

При $n = 1$ получаем сверточный слой $1 \times 1 \times K$ ценой возврата к линейной модели. Возможности:

- произвольное изменение глубины feature map;
- агрегация коррелированных признаков и избавление от слабых признаков.

Inception

Проблема: определение оптимального размера ядра. Решение: использование разных сверток и пулинга и конкатенация выходов в общую feature map. Свертки 1×1 уменьшают размерность для ускорения вычислений.

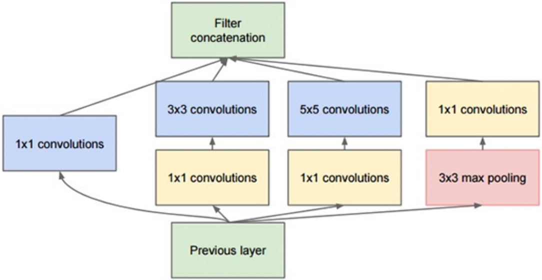

Рис.: Архитектура блока Inception

Архитектура GoogLeNet

Архитектура GoogLeNet (2014). Параметры: 9 блоков Inception, 2 вспомогательных классификатора. Вход: 224×224, выход: 1000 классов.

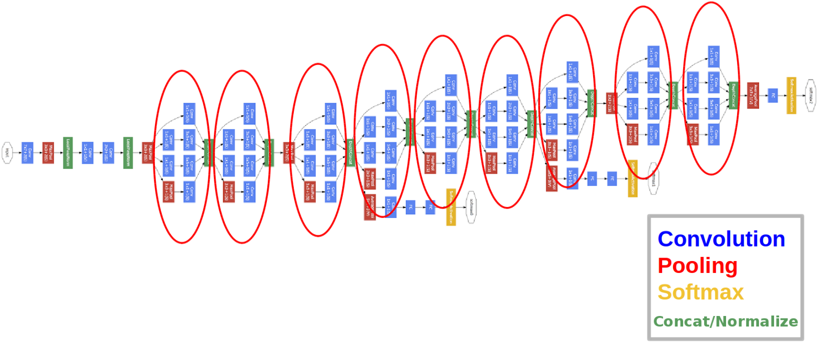

Рис.: Архитектура GoogLeNet

Внутренний ковариационный сдвиг

Общая задача классификации с учителем:

$$q(x|y) = p(y|x;\theta).$$

На некоторой выборке (X, Y) оптимизируем $\hat{\theta}$:

$$\hat{\theta} = \arg\min Q(Y, \mathrm{E}[Y|X;\theta]).$$

Для многослойной сети $x = \sigma(f^{(s-1)}), y = f^{(s)}, \hat{\theta} = W^{(s)}$. С увеличением количества шагов градиента распределение первого слоя стабилизируется, т.к. $\hat{\theta} \to \theta$. Однако вход второго слоя остается нестабилен до тех пор, пока

$$|\theta - \hat{\theta}| > \epsilon$$

для некоторого $\epsilon > 0$. Эта проблема носит название внутреннего ковариационного сдвига. Способ решения: стандартизация входов.

Batch normalization

Пусть:

- $f(i,j) : Z \times Z \to R^K$ — feature map на входе слоя глубины K и размерности $N \times M$;
- $x = x_{i,j} \in R^K$ – входные индивиды (пиксели).

Предполагаем:

$$x \sim P(x; \Sigma, \mu) \quad (K - \text{мерная.})$$

Пусть $X \in R^{K \times MN}$ – набор всех индивидов на $f(i,j)$. Тогда $\hat{\Sigma}$ — оценка матрицы ковариации входа, $\hat{\mu}$ — оценка смещения входа. Пересчет градиента такой модели требует вычисления:

$$\frac{\partial}{\partial x} \hat{\Sigma}^{-1/2}(x - \hat{\mu}).$$

Batch normalization

Предположим, что компоненты независимы, т.е. для $P(x; \Sigma, \mu)$ Σ диагональная:

$$\Sigma = diag(\sigma_k^2).$$

Стандартизуем покомпонентно:

$$\bar{x}_{k,i} = \frac{x_{k,i} - \hat{\mu}_k}{\hat{\sigma}_k}.$$

Проблема: ограниченность области значений $\bar{x}$. Решение: ввод линейной комбинации с обучаемыми параметрами:

$$y_{k,i} = \gamma_k \bar{x}_{k,i} + \beta_k.$$

Процедура BN сводится к удалению случайных параметров распределения входных данных и замене их на обучаемые.

Batch normalization

Отказ от смещения b в модели нейрона ничего не изменит, но уменьшит количество параметров.

Возможности инструмента:

- регуляризация за счет исправления смещения;
- общее ускорение обучения.

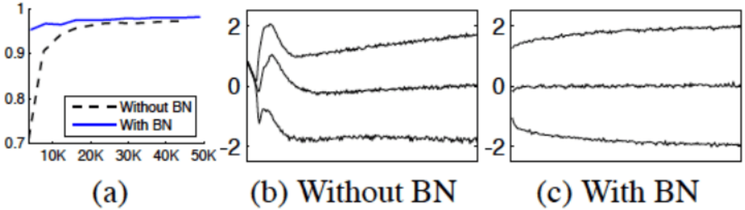

Рис.: (a): изменение скорости обучения при использовании bn; (b, c): 15, 50 и 85-квантили для выходов одного из слоев в зависимости от эпохи

Деградация сети

Деградация сети: при увеличении количества слоев нейросети интуитивно логично ожидать улучшения качества предсказания, что в общем случае не так. Предполагаемые причины: затухание градиента.

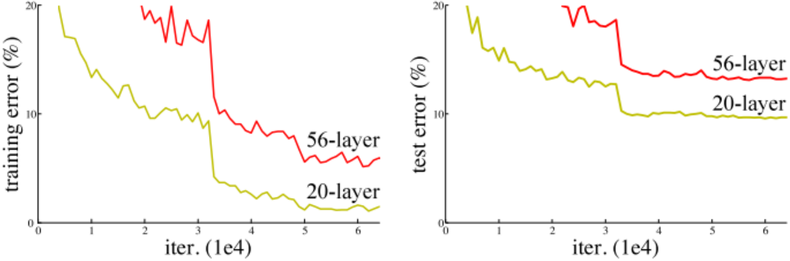

Рис.: Ошибка на тренировочной (a) и тестовой (b) выборках сети с 20 и 56 слоев

Идея ResNet

Решение как в бустинге: обучаться на остатках. Пусть аппроксимируем $\mathcal{H}(x)$ сетью $\mathcal{F}(x)$; предлагается использовать модель вида $\mathcal{H}(x) = \mathcal{F}(x) - x$.

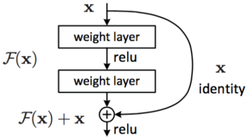

Рис.: Модель слоя с подключением ко входу

Inception ResNet v2

Архитектура сети Inception ResNet v2 (2016). Используются различные повторяющиеся блоки Inception с подключением выхода для вычисления остатков.

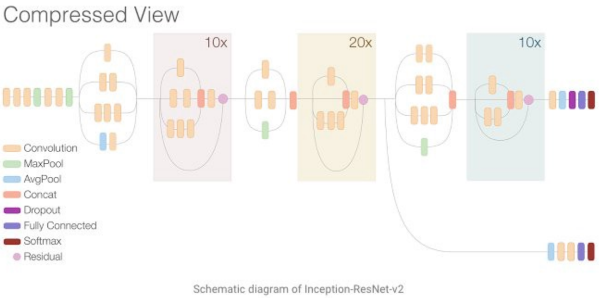

Рис.: Архитектура сети Inception ResNet v2

DenseNet

Изменен вид целевой функции алгоритма: на вход к текущему слою подаются выходы всех предыдущих:

$$x^{(s)} = \mathcal{F}(x^{(1)}, x^{(2)}, \dots, x^{(s-1)}),$$

Глубина последнего слоя: $K^{(M)} = K^{(0)} + \sum_{s=1}^{M} K^{(s)}$, где $K^{(0)}$ — глубина входного слоя, $K^{(s)}$ — глубина слоя s. Предлагается использовать неглубокие слои с малыми ядрами.

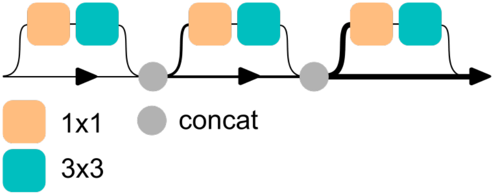

Рис.: Схема блока DenseNet на трех слоях

DenseNet

Архитектура сети DenseNet (2017). Используются различные повторяющиеся блоки Inception с подключением выхода для вычисления остатков.
Параметры: 4 «Dense» блока: на 6, 12, 24 и 16 слоев; переходные слои для уменьшения размерности. Вход 112×112, выход 1000 классов через полносвязный слой.

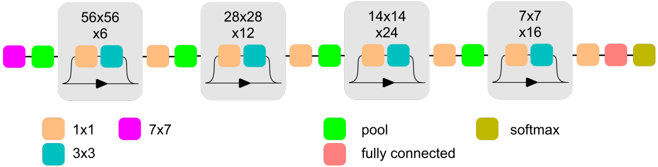

Рис.: Архитектура сети DenseNet

Быстрее обучается и точнее работает по сравнению с ResNet с аналогичным количеством параметров.